

DEPARTMENT OF CIVIL ENGINEERING
COLLEGE OF ENGINEERING
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23508

Langley SPAN
IN-39-CR
90668
P-129

**SUBSTRUCTURE ANALYSIS USING NICE/SPAR AND APPLICATIONS
OF FORCE TO LINEAR AND NONLINEAR STRUCTURES**

By

Zia Razzaq, Principal Investigator

Venkatesh Prasad, Graduate Student

Siva Prasad Darbhamulla, Graduate Student

Ravinder Bhati, Graduate Student and

Cai Lin, Graduate Student

Progress Report
For the period ended June 30, 1987

Prepared for the
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Under
Research Grant NAG-1-438
Dr. Olaf O. Storaasli, and
Dr. W. Jefferson Stroud, Technical Monitors
SSD-Structural Mechanics Branch

(NASA-CR-180317) SUBSTRUCTURE ANALYSIS
USING NICE/SPAR AND APPLICATIONS OF FORCE TO
LINEAR AND NONLINEAR STRUCTURES Progress
Report, period ending 30 Jun. 1987 (Old
Dominion Univ.) 129 p Avail: NTIS HC

N87-27260

Unclas
G3/39 0090668

August 1987

DEPARTMENT OF CIVIL ENGINEERING
COLLEGE OF ENGINEERING
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23508

**SUBSTRUCTURE ANALYSIS USING NICE/SPAR AND APPLICATIONS
OF FORCE TO LINEAR AND NONLINEAR STRUCTURES**

By

Zia Razzaq, Principal Investigator

Venkatesh Prasad, Graduate Student

Siva Prasad Darbhamulla, Graduate Student

Ravinder Bhati, Graduate Student and

Cai Lin, Graduate Student

Progress Report
For the period ended June 30, 1987

Prepared for the
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Under
Research Grant NAG-1-438
Dr. Olaf O. Storaasli, and
Dr. W. Jefferson Stroud, Technical Monitors
SSD-Structural Mechanics Branch

Submitted by the
Old Dominion University Research Foundation
P. O. Box 6369
Norfolk, Virginia 23508

August 1987

ACKNOWLEDGMENTS

The technical help and moral support provided by Dr. Olaf O. Storaasli and Dr. W. Jefferson Stroud as well as their discussions have been of tremendous value. The interaction with Jonathan B. Ransom for substructure analysis using NICE/SPAR, and Sue Bostic for on-line plotting of speedup curves, greatly facilitated the work. Thanks are due to Dr. Harry Jordan and Norbert Arenstorf of the University of Colorado for the help and interaction provided with regard to FORCE.

SUBSTRUCTURE ANALYSIS USING NICE/SPAR AND APPLICATIONS
OF FORCE TO LINEAR AND NONLINEAR STRUCTURES

By

Zia Razzaq¹, Venkatesh Prasad², Siva Prasad Darbhamulla²,
Ravinder Bhati², and Cai Lin²

ABSTRACT

Parallel computing studies are presented for a variety of structural analysis problems. Included are the substructure planar analysis of rectangular panels with and without a hole, the static analysis of space mast, using NICE/SPAR and FORCE, and substructure analysis of plane rigid-jointed frames using FORCE. The computations are carried out on the Flex/32 MultiComputer using one to eighteen processors. The NICE/SPAR runstream samples are documented for the panel problem. For the substructure analysis of plane frames, a computer program is developed to demonstrate the effectiveness of a substructuring technique when FORCE is enforced. On-going research activities for an elasto-plastic stability analysis problem using FORCE, and stability analysis of the focus problem using NICE/SPAR are briefly summarized. Speedup curves for the panel, the mast, and the frame problems provide a basic understanding of the effectiveness of parallel computing procedures utilized or developed, within the domain of the parameters considered. Although the speedup curves obtained exhibit various levels of computational efficiency, they clearly demonstrate the excellent promise which parallel computing holds for the structural analysis problems.

¹ Associate Professor, Department of Civil Engineering, Old Dominion University, Norfolk, Virginia 23529.

² Graduate Student, Department of Civil Engineering, Old Dominion University, Norfolk, Virginia 23529.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	ii
ABSTRACT	iii
1. SUBSTRUCTURE ANALYSIS OF RECTANGULAR PANEL WITH HOLE	
USING NICE/SPAR AND FORCE.....	1
1.1 Problem Description	1
1.2 Finite Element Description and Substructuring	1
1.3 Solution Procedure and Parallel Computation	2
1.4 Numerical Results and Discussion	3
2. CONCURRENT STATIC ANALYSIS OF MAST USING FORCE.....	5
2.1 The Mast Problem	5
2.1 Results and Discussion	5
3. ELASTO-PLASTIC STABILITY ANALYSIS USING FORCE.....	6
3.1 Problem Definition and Goals	6
3.2 Concurrent Elasto-plastic Algorithm Using FORCE	6
3.3 Publication	7
4. SUBSTRUCTURE ANALYSIS OF PLANE FRAMES USING FORCE.....	8
4.1 Problem Definition and Goals	8
4.2 Substructuring Technique	8
4.3 Concurrent Algorithm	10
4.4 Numerical Results for 3-Story Frame	12
5. BUCKLING ANALYSIS OF FOCUS PROBLEM USING NICE/SPAR	15
TABLES	16
FIGURES	22
REFERENCES	35

	Page
APPENDIX A: RELEVANT SPAR PROCESSORS AND CLAMP PROCEDURES	36
APPENDIX B: NICE/SPAR RUNSTREAM SAMPLES	40
APPENDIX C: PROGRAM FOR ELASTO-PLASTIC STABILITY PROBLEM	68
APPENDIX D: PROOF-COPY OF THE CONCURRENT PROCESSING PAPER	
FROM ENGINEERING WITH COMPUTERS JOURNAL.....	99
APPENDIX E: FRAME SUBSTRUCTURING.....	109
APPENDIX F: FORCE PROGRAM LISTING WITH SAMPLE INPUT/OUTPUT	
FOR SUBSTRUCTURE ANALYSIS OF 3-STORY FRAME.....	113
APPENDIX G: TITLES OF ABSTRACTS FOR 29TH AIAA/SDM CONFERENCE	132

1. SUBSTRUCTURE ANALYSIS OF RECTANGULAR PANEL WITH HOLE

USING NICE/SPAR AND FORCE

1.1 Problem Description

A two-dimensional elastic stress analysis of a rectangular panel with a central hole is conducted using NICE/SPAR while invoking concurrent processing with the use of FORCE. References 1 through 5 represent the relevant documents on NICE/SPAR and FORCE. Figure 1 shows the rectangular panel dimensions and the applied uniform compressive loading. The Young's modulus and the Poisson's ratio of the panel material are, respectively 10,000 ksi, and 0.3, which represent a typical aluminum alloy. The panel thickness is taken as 0.1 in. Initially, the panel is analyzed in the absence of a central hole using substructuring. The analysis of panel including the hole is then conducted using two substructures representing the regions around and away from the hole.

1.2 Finite Element Discretization and Substructuring

Figures 2 through 4 show the various types of discretizations investigated using E41 quadrilateral elements of NICE/SPAR. Figure 2(a) shows the rectangular panel divided into four elements as well as the boundary simulations. Figure 2(b) shows substructures 1 and 2 with revised element and node numbering at the substructure level. Similarly Figure 3 shows an eight-element discretization and the corresponding substructures. Figure 4 shows the finite element discretization of the rectangular panel with hole. Figure 4(a) shows the node numbering scheme commencing from the inner region around the hole for nodes 1 through 24, and continues at the bottom left corner of the panel with nodes 25

through 78. Figure 4(b) shows the element numbering scheme commencing from the inner region around the hole for elements 1 through 16, and for the remainder of the panel for elements 1 through 44 starting at the bottom left corner. The common element numbering from 1 to 16 can be used as long as the outer and the inner regions of the panel are modelled as separate substructures as shown in Figures 5(a), and 5(b), respectively.

1.3 Solution Procedure and Parallel Computations

The solution procedure for the substructuring method using NICE/SPAR is given in this section. The relevant SPAR processors and CLAMP procedures. The steps of the procedure are as follows:

1. The AUS subprocessor TABLE is created to input the number of substructures, the number of nodes per element, and the number of elements in each substructure.
2. Interface nodes are entered using subprocessor TABLE. These are stored in the NICE/SPAR library in the form of data sets.
3. Steps 1 and 2 are stored in a procedure called SUB_TABLE.
4. An external file PRESERVE is introduced into the main runstream. This file consists of CLAMP procedures END_SUB and GLOB_DAT.
5. The individual substructure data such as the number of nodes in the substructure, material properties, joint locations, boundary conditions, and element connectivity are entered. At the end of each substructure data, procedure END_SUB is called to store the data set sequence number.
6. Procedure GLOB_DAT is called for generating the data sets required for the assembled structure, from the data obtained from each

substructure.

7. Processor K is executed to obtain the final stiffness matrix.
8. subroutine getrow is used to extract the K matrix from the NICE/SPAR data library, and stored by rows in the upper triangular part of the matrix.
9. The FORCE subroutine SGEFA is called to perform the decomposition of the square matrix using gaussian elimination with partial pivoting.
10. Subroutine SGESL is called to solve the system of equations for the static displacements by back substitution.
11. The FORCE subroutines in steps 9 to 10 are run on several processors for speedup study.

1.4 Numerical Results and Discussion

In this section as well as sections 2 and 4 of this report, the following definitions of speedup and efficiency are used. Speedup is defined as the execution time on a uniprocessor divided by execution time on n processors. Parallel efficiency is defined as the speedup divided by the number of processors times 100. Table 1 presents the speedup and efficiencies for the four element rectangular panel shown in Figure 2. The corresponding speedup versus the number of processors relationship is shown in Figure 6, involving a 27×27 stiffness matrix. The maximum speedup factor is 3.7 for 8 processors. The efficiency reduces continuously with an increase in the number of processors.

Table 2 presents the speedup and efficiencies for the eight-element rectangular panel shown in Figure 3. The corresponding speedup versus number of processors relationship is shown in Figure 7, involving a 45×45 stiffness matrix. The maximum speedup factor is 5.14 for 8 and 12

processors. The efficiency reduces continuously with an increase in the number of processors, though at a lower rate than that for the four element panel.

Table 3 presents the speedup and efficiencies for the rectangular panel with a central hole shown in Figure 4. The corresponding speedup versus the number of processors relationship is in Figure 8, involving a 234×234 stiffness matrix. The maximum speedup factor is 13.2 for 18 processors. Here the speedup increases continuously with an increase in the number of processors while the efficiency decreases gradually.

2. CONCURRENT STATIC ANALYSIS OF MAST USING FORCE

2.1 The Mast Problem

Figure 9 shows the mast problem. The mast is a 60-meter high truss divided into 55 platforms with three nodes at each platform level. Each node has three degrees of freedom, that is, displacements in x, y, and z directions. The mast consists of horizontal batten members, inclined diagonal members, and vertical longeron members. The bottom platform with first three nodes is constrained against displacements. A concentrated load of 100 tons is applied at the upper most node in the global x direction. The datasets were stored in the NICE/SPAR library named MASTM3.L01. The problem is to determine the static nodal deflections using NICE/SPAR and FORCE.

2.2 Results and Discussion

The stiffness matrix generated by NICE/SPAR is of the order 495 X 495, including the support nodes. The linear system of equations is solved using FORCE. Table 4 presents the speedup and efficiencies when 1, 2, 4, 8, 12, 16, and 18, processors are used to solve the system of equations. The highest speedup is obtained when 18 processors are used, and is 15.71. The computational efficiency decreases slightly with an increase in number of processors. The results are shown by the dashed curve in Figure 10, in which the theoretical ideal relationship is also shown for a direct comparison. In comparison to the results presented in Section 1 for the rectangular panel problem of the order 234 X 234, the speedups for the mast show better promise due to an increase of the stiffness matrix size to 495 X 495.

3. ELASTO-PLASTIC STABILITY ANALYSIS USING FORCE

3.1 Biaxially Restrained Imperfect Column

In a study reported in Reference 6 by Darbhamulla, Razzaq, and Storaasli, a concurrent elasto-plastic solution for the stability analysis of biaxially restrained imperfect column was presented which employed the Finite Element Machine. Appendix C presents the listing of the corresponding computer program in PASCAL. This program is being converted to CONCURRENT FORTRAN so that the efficiency of the algorithm can be studied using FORCE developed by Jordan and Norbert (Reference 5) and implemented on FLEX/32. As described in Reference 6, the problem involves solving three coupled nonlinear differential equations whose coefficients vary with the applied loads and spatial coordinates. In that sense, the concepts developed are of general use in various types of materially nonlinear structural problems. A brief outline of the algorithm utilizing FORCE is given in the following section.

3.2 Concurrent Elasto-Plastic Algorithm Using FORCE

The computer program given in Appendix C is being converted to CONCURRENT FORTRAN based on the following algorithm:

1. Read initial data and compute required cross-sectional properties.
2. Assemble a global stiffness matrix and a force vector concurrently using FORCE.
3. Compute initial displacement vector using the FORCE simultaneous equation solver.
4. Synchronize all processors.
5. Compute elasto-plastic cross-sectional properties at various

locations along the length concurrently using the procedure presented in Reference 8.

6. Reassemble the global stiffness matrix and the new force vector concurrently, and update the displacement vector using, again, the FORCE simultaneous equation solver.
7. Check convergence and proceed to Step 8 if convergence is achieved. Otherwise, go to Step 4.
8. Increment the external load and go to Step 4 if the member collapse has not occurred.

The program is being implemented on the FLEX/32 MMOS computer.

3.3 Publication

Reference 8 is scheduled to appear in Engineering with Computers, An International Journal for Computer-aided Mechanical and Structural Engineering this year. The proof-copy of this paper is given in Appendix D.

4. SUBSTRUCTURE ANALYSIS OF PLANE FRAMES USING FORCE

4.1 Problem Definition and Goals

The problem is to develop a concurrent algorithm for the linear load-deflection analysis of rigid-jointed plane frames using substructuring. Figure 11 (a) shows an example of a three story frame ABCDEFGH with a span L and story height h. Figure 11 (b) shows one way of dividing the structure into two substructures, CDEF and ABCFGH. The main goal of this study is to evaluate the computational efficiency of the concurrent algorithm developed while employing FORCE when substructures of the type shown in Figure 11 are adopted. For the present, a special purpose matrix stiffness substructure analysis program has been developed and implemented on FLEX/32 parallel computer for a direct investigation of the effectiveness of FORCE, as the main objective. The concurrent substructuring technique being considered may be generalized for arbitrary frames for implementation on NICE/SPAR later.

4.2 Substructuring Technique

The substructuring technique adopted for the analysis of the frame considered here is given in Reference 9 and summarized in Appendix E of this report. A number of equations presented in Appendix E are used in a concurrent algorithm outlined in Section 4.3. The following is a brief summary of the applicable equations. The various terms are defined in Appendix E.

The static joint equilibrium matrix equation for the k-th substructure is given by Equation E.1 as follows:

$$[S]\{\Delta\} = \{P\} + \{R\} \quad (1)$$

$$[S_{uu}]_k \{\Delta_u\}_k = \{P_u\}_k, \quad (2)$$

for $k = 1, 2, 3, \dots, n$. The reactions at the restrained nodes can be found using Equation E.5:

since $\{P_r\} = \{0\}$ for the substructures shown in Figure 11 (b) owing to the fact that the nodes C, F, A and H, are completely restrained.

$$[S_s]_k = [S_{rr}]_k - [S_{ru}]_k [S_{uu}]_k^{-1} [S_{ur}]_k \quad (4)$$
$$[\lambda_{ur}]_k = [s_{uu}]_k^T [s_{ur}]_k \quad (5)$$
$$[S_{uu}]_k [\lambda_{ur}]_k = [S_{ur}]_k \quad (6)$$
$$[\lambda_{ur}]_k = [\{\lambda_{u1}\} \ \{\lambda_{u2}\} \dots \{\lambda_{ur}\}]_k \quad (7)$$

and $\{ \}$ are column vectors of the order $u \times 1$. Thus, Equation 6 can be written as a series of matrix equations as follows:

$$[S_{uu}]_k \{\lambda_{ur}\}_k = \{S_{ur}\}_k$$

9

indicated in Eq. 4 without inverting the matrix $[S_{uu}]_k$. Equation 4 can now be written as:

$$[S_s]_k = [S_{rr}]_k - [S_{ru}]_k [\lambda_{ur}]_k \quad (10)$$

The static equilibrium equation for the nodes linking the various substructures is given by Equation E.13 as follows:

$$[S_a] \{\Delta_a\} = \{P_a\} + [R_a], \quad (11)$$

in which the various terms of $[S_a]$ and $\{P_a\}$ are obtained as explained in Section E.3 of Appendix E.

The true deflection vector, $\{\Delta_{rd}\}$, for the nodes linking the various substructures is found using Equation E.16 in the following form:

$$\{P_{rd}\} = [S_{ll}] \{\Delta_{rd}\} \quad (12)$$

The deflections of the unrestrained nodes for the k-th substructure is found using Equation E.19.

$$\{\Delta_{un}\}_k = (\{\Delta_u\} - [S_{ur}] \{\Delta_{rd}\})_k \quad (13)$$

The procedure described in this section can be readily applied to any type of framed structures behaving in a linear elastic manner. A concurrent algorithm utilizing this technique is given in the following section.

4.3 Concurrent Algorithm

Based on the substructuring technique outlined in Section 3.2, the following concurrent algorithm has been developed on FLEX/32 parallel computer.

1. Formulate the stiffness matrix $[S]$ given in Equation 1 for the substructures $k = 1, 2, 3, \dots, m$, on n processors concurrently.
2. Extract $[S_{uu}]_k$ appearing in Equation 2, from the $[S]$ matrices, concurrently, for all k values.

3. Solve Equation 2 for $\{\Delta_u\}_k$ with $k = 1$, to determine the deflections of Substructure 1 by invoking FORCE simultaneous equations solver SOLVE, concurrently using n processors. Similarly, solve Equation 2 for $\{\Delta_u\}_k$ with $k = 2, 3, \dots, m$, corresponding to Substructures 2, 3, ..., m , using FORCE.
4. Extract $[S_{ru}]_k$ appearing in Equation 3, from the $[S]$ matrices, concurrently for all k values.
5. Compute the vectors $\{R_r\}_k$ with $k = 1$ to determine the reactions in Substructure 1 by invoking FORCE matrix multiplication subroutine called MATMP, concurrently. Similarly, determine $\{R_r\}_k$ for $k = 2, 3, \dots, m$, corresponding to Substructures 2, 3, ..., m , using FORCE.
6. For $k = 1$, determine $\{\lambda_{uv}\}_k$ with $v = 1$ as represented in Equations 9 using SOLVE, concurrently on n processors. Repeat the process of finding $\{\lambda_{uv}\}_k$ for $v = 2, 3, \dots, r$, respectively, to generate the matrix $[\lambda_{ur}]_k$ as indicated in Equation 7. Obtain $[S_s]_k$ using Equation 10 concurrently on n processors.
7. Repeat Step 6 for $k = 2, 3, \dots, m$.
8. Assemble the global matrix $[S_a]$ and the vector $\{P_a\}$ appearing in Equation 11, on a single processor using the $[S_s]_k$ matrices generated in Steps 6 and 7; $\{R_r\}_k$ generated in Step 5, while employing Equations E.14 and E.16 given in Appendix E.
9. From $[S_a]$ and $\{P_a\}$ obtained in Step 8, formulate Equation 12.
10. Solve Equation 12 concurrently, using SOLVE, on n processors.
11. For $k = 1$, determine the true deflection $\{\Delta_{un}\}_k$ of the unrestrained nodes, using Equation 13, concurrently on n processors.
12. Repeat Step 11 for $k = 2, 3, \dots, m$.

The algorithm outlined above is coded in CONCURRENT FORTRAN for the problem shown in Figure 11. A listing of the program is given in Appendix F with a sample input/output. The computational efficiency of the algorithm is explained in the following section.

4.4 Numerical Results for 3-Story Frame

With three degrees-of-freedom per joint (vertical and horizontal displacements, and a rotation), the frame stiffness matrix is of the order 18×18 . Table 5 presents the computational time t , speedup factor s , and efficiency for 1 to 5 parallel processors for a concurrent analysis of the 3-story frame. The time t is measured in rtick units. One rtick equals 1/50th of a second. For solving the 18 simultaneous equations concurrently, Subroutine SOLVE developed by Jordan and his research associates from the University of Colorado, was utilized. A complete listing of the subroutine appears as a part of the computer program presented in Appendix F. The speedup factor versus the number of processors data given in Table 5 is used to plot the 'actual' curve in Figure 12 in which the theoretical ideal relationship is also given for a comparison. The maximum speedup is 2.60 and is obtained when a total of 4 processors are used corresponding to a computational efficiency of 65%. The use of 2 processors, however, results in an efficiency of 81%.

Table 6 presents the computational times T_1 through T_4 , defined as follows:

T_1 = time taken to assemble the stiffness matrices $[S]$ given in Equation 1, for Substructures 1 and 2 shown in Figure 11,

T_2 = time taken for solving Equations 2, 3 and 4 for Substructure 1 with each of the matrices S_{uu} , S_{ru} and S_s of order 6×6

T_3 = time taken for solving Equations 2, 3 and 4 for Substructure 2, with the matrices S_{uu} , S_{ru} and S_s of the order 6×6 , 12×6 and 12×12 respectively,

T_4 = time taken for solving Equation 12 for the complete system;

the speedup factor, s , based on the total time T_t given by:

$$T_t = T_1 + T_2 + T_3 + T_4;$$

and efficiency η based on s ; for 1 to 5 parallel processors for a concurrent substructure analysis of the 3-story frame. The FORCE subroutine SOLVE was used concurrently for the two substructures for obtaining solutions to a pair of six simultaneous equations. It is noted here that there are two levels of computational parallelization involved in this solution scheme:

Level 1: parallel generation of the two substructure stiffness matrices on two different processors, and

Level 2: parallel solution of two sets of six simultaneous equations using SOLVE on up to five processors.

The parallel processors work on one set of equations at a time.

The value of T_1 ranges from 3 to 5 rticks. The T_2 value is in the range from 3 to 4 rticks, whereas T_3 is in the range from 5 to 10 rticks. One may expect to obtain nearly identical values for T_2 and T_3 since they represent the computational times for handling the two substructures with equal number of free nodes. That this is not indeed the case can be understood by realizing that the orders of S_{ru} and S_s matrices for Substructures 1 and 2 are not equal, as described above in the definitions for T_2 and T_3 . The value of T_4 is halved when 2 to 5 processors are used instead of just one. The total time T_t varies between 12 and 21 rticks, the latter being the time taken by one

processor. For 2, 3 and 4 processors, T_t value is nearly constant. For 5 processors it is greater than for 2, 3 or 4 processors. The maximum speedup is 1.75 and is obtained when a total of 3 or 4 processors are used. However, the efficiency with 3 processors is greater than that with 4 processors. The maximum efficiency is obtained with 2 processors and is 80.5%, of course excluding the case of one processor.

The reason for the relatively low speedups obtained is that the matrix sizes for the problem considered are relatively small. As the number of processors is increased, the overhead time (for example, while executing the Barrier, End Barrier and Join statements) becomes significant as compared to the time consumed by arithmetical operations required for the simultaneous equations.

5. BUCKLING ANALYSIS OF FOCUS PROBLEM USING NICE/SPAR

The buckling solution to the CSM Focus Problem 1 outlined in Reference 2 is being run for various discretizations. The finite element meshes are generated using CSM1 with NICE/SPAR E43 finite elements. Presently, all computations are being carried out on a single FLEX/32 processor. The implementation of concurrent buckling and postbuckling solution procedures will be carried out once rigorous study of these problems is completed on a single processor.

Table 1. Speedup and efficiencies for four-element rectangular panel

Number of Processors	Computational time (secs)	Speedup	Efficiency
1	0.96	-	-
2	0.54	1.78	89.0
4	0.28	3.43	85.7
8	0.26	3.70	46.2
12	0.30	3.20	26.7
16	0.42	2.29	14.3
18	0.48	2.00	12.0

Table 2. Speedup and efficiencies for eight-element rectangular panel

Number of Processors	Computational time (secs)	Speedup	Efficiency
1	3.60	-	-
2	1.88	1.91	95.7
4	1.06	2.25	56.2
8	0.70	5.14	64.3
12	0.70	5.14	42.8
16	0.84	4.30	27.0
18	0.92	3.91	21.7

Table 3. Speedup and efficiencies for rectangular panel with central hole

Number of Processors	Computational time (secs)	Speedup	Efficiency
1	474.62	-	-
2	248.74	1.90	95.4
4	128.82	3.69	92.1
8	66.80	7.10	88.8
12	47.50	9.99	83.3
16	38.56	12.31	77.0
18	35.96	13.20	73.3

Table 4. Speedup and efficiencies for the mast problem

Number of Processors	Computational time (secs)	Speedup	Efficiency
1	4530.02	-	-
2	2388.80	1.90	94.8
4	1228.10	3.69	92.3
8	610.00	7.43	93.0
12	417.38	10.85	90.5
16	320.60	14.13	88.3
18	288.30	15.71	87.3

Table 5. Computational time measurements, speedup factors and efficiencies for concurrent analysis of 3-story frame

Number of Processors	Execution time t (rticks)	Speedup factor	Efficiency η
1	13	1.00	100.0
2	8	1.62	80.5
3	6	2.16	58.3
4	5	2.60	43.7
5	6	2.16	28.0

Table 6. Computational time measurements, speedup factors, and efficiencies for concurrent substructure analysis of 3-story frame

Number of Processors	Execution time t (rticks)				Total	Speedup	Efficiency
	T_1	T_2	T_3	T_4	Time	Factor	η
1	5	4	10	2	21	1.00	100.0
2	3	3	6	1	13	1.62	80.5
3	3	3	5	1	12	1.75	58.3
4	3	3	5	1	12	1.75	43.7
5	4	4	6	1	15	1.40	28.0

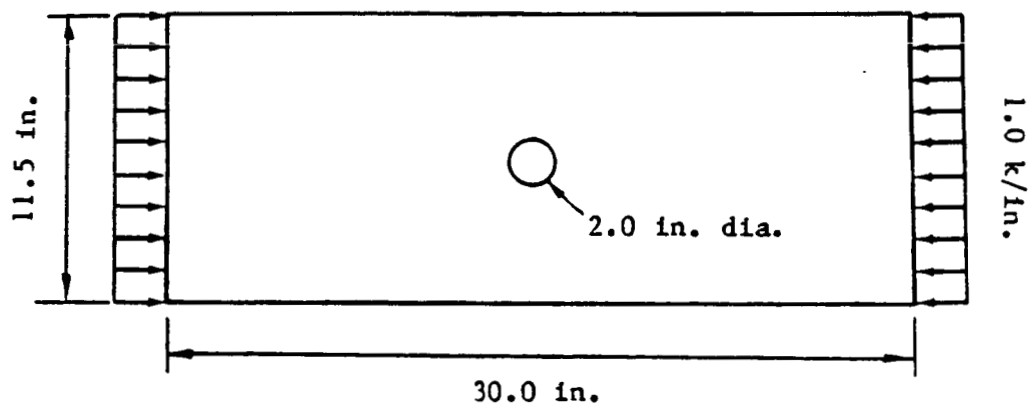
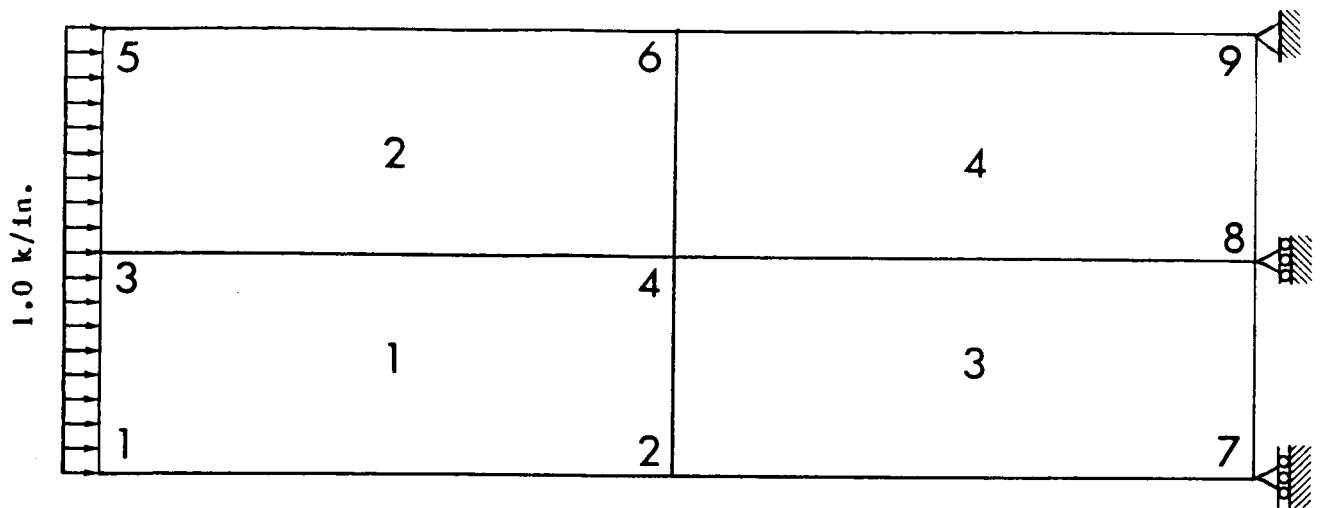
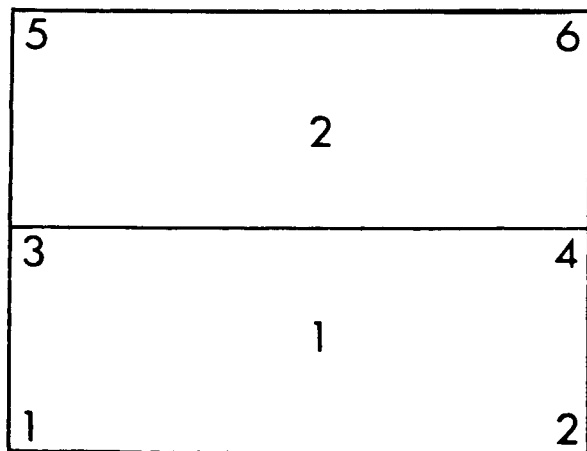


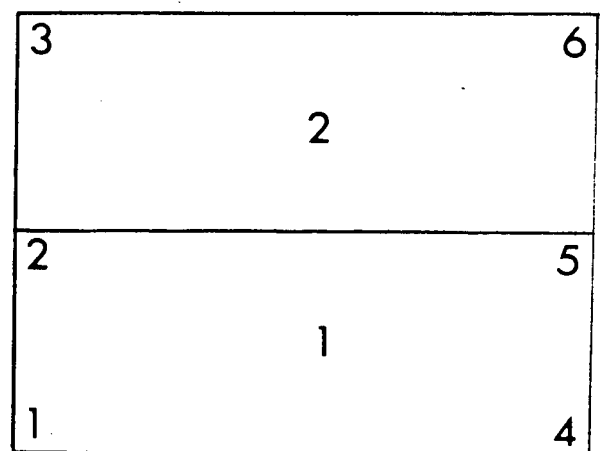
Figure 1. Rectangular panel with hole and uniform compression



(a)



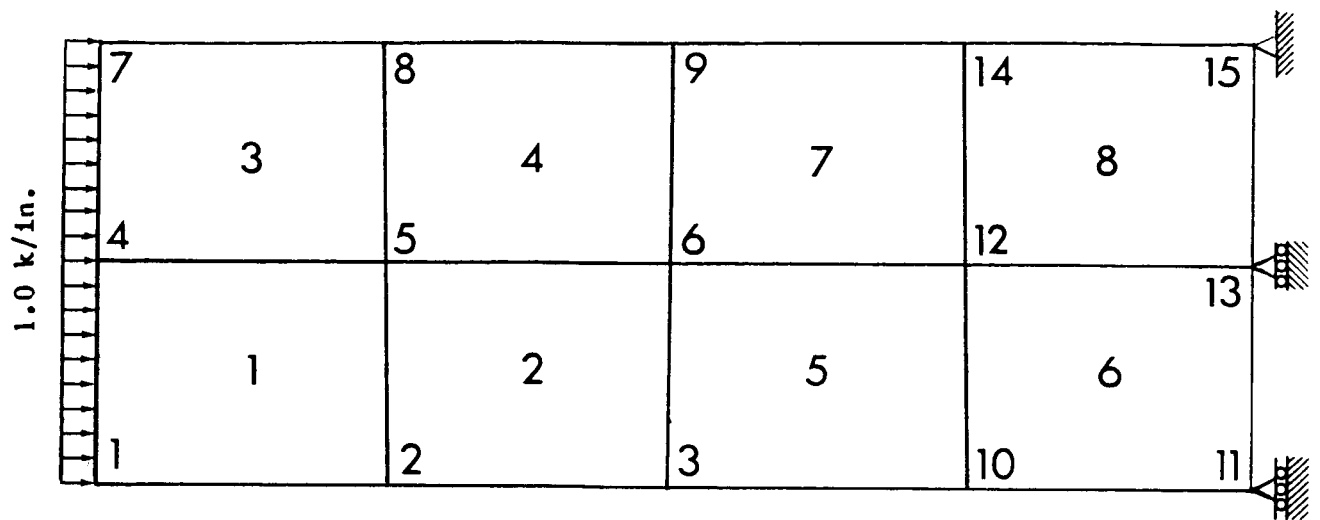
Substructure 1



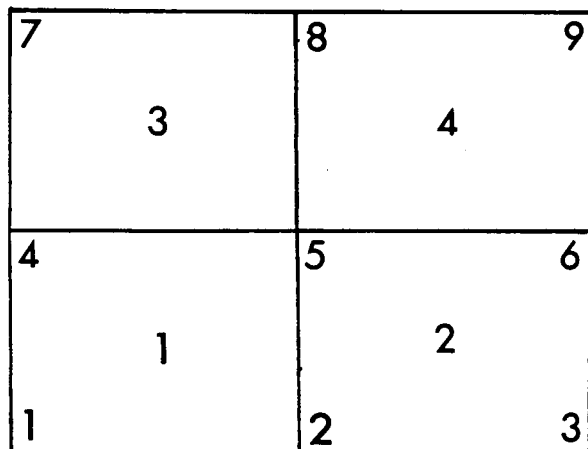
Substructure 2

(b)

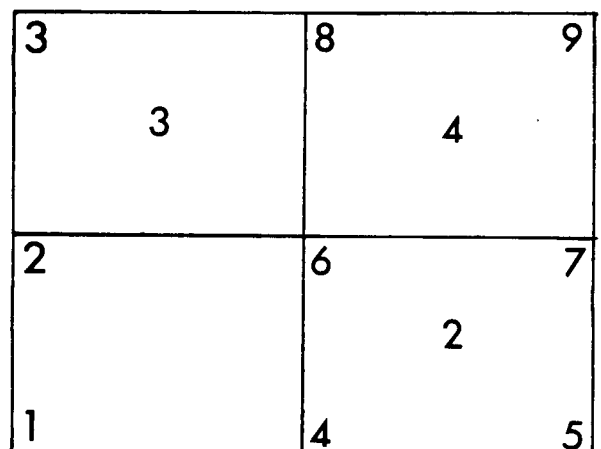
Figure 2. Substructuring of panel with four elements



(a)



Substructure 1



Substructure 2

(b)

Figure 3. Substructuring of panel with eight elements

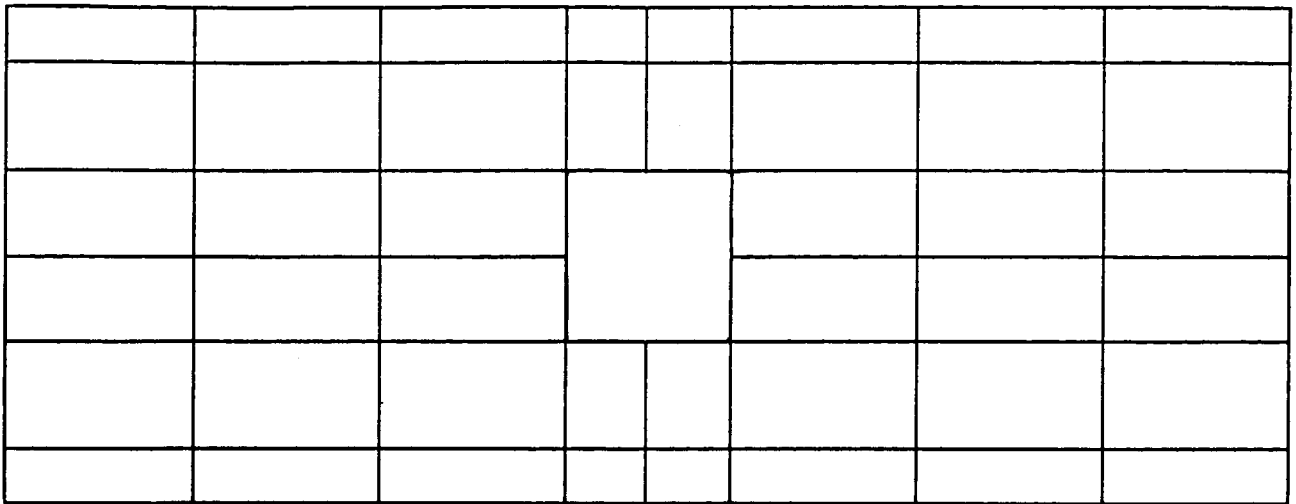
70	71	72	73	74	75	76	77	78
61	62	63	64	65	66	67	68	69
55	56	57	20	19	18	58	59	60
49	50	51	12	11	10	52	53	54
43	44	45	21	9	17	46	47	48
34	35	36	37	38	39	40	41	42
25	26	27	28	29	30	31	32	33

(a) Node numbering scheme

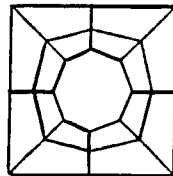
1.0 k/in.	37	38	39	40	41	42	43	44
	29	30	31	32	33	34	35	36
	23	24	25	11	10	26	27	28
	17	18	19	12	9	20	21	22
	9	10	11	13	8	14	15	16
	1	2	3	4	5	6	7	8

(b) Element numbering scheme

Figure 4. Finite element discretization of panel with hole



(a) Substructure 1



(b) Substructure 2

Figure 5. Substructuring of panel with hole and sixty elements

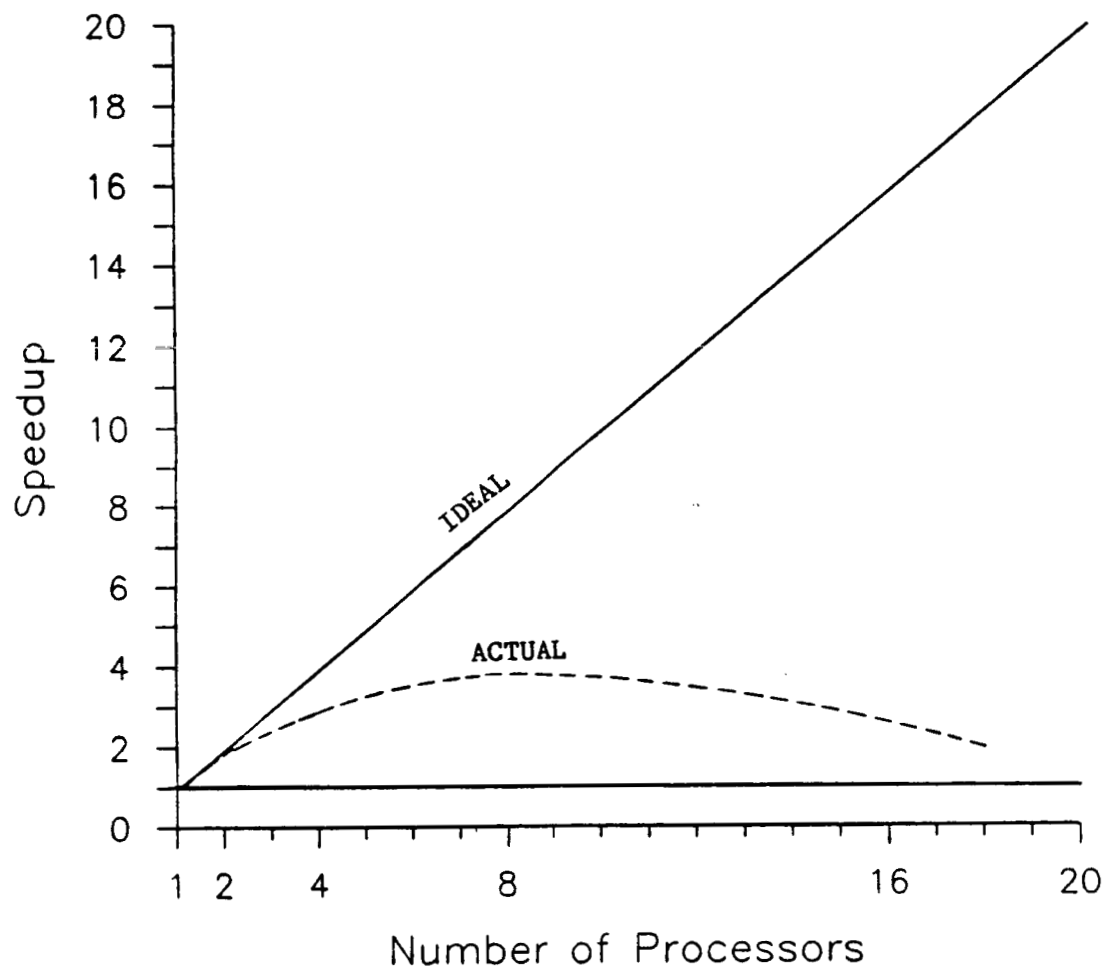


Figure 6. Speedup factor versus number of processors for four-element rectangular panel

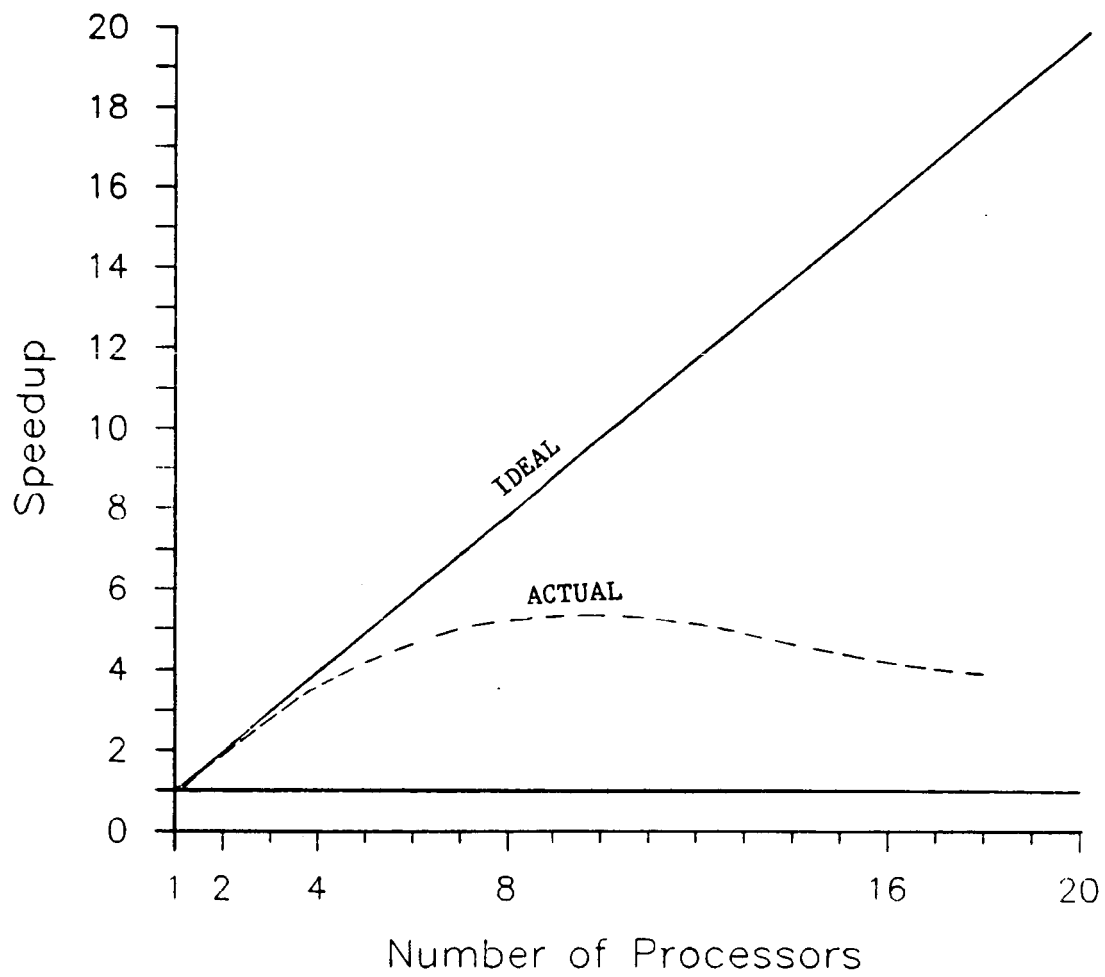


Figure 7. Speedup factor versus number of processors for eight-element rectangular panel

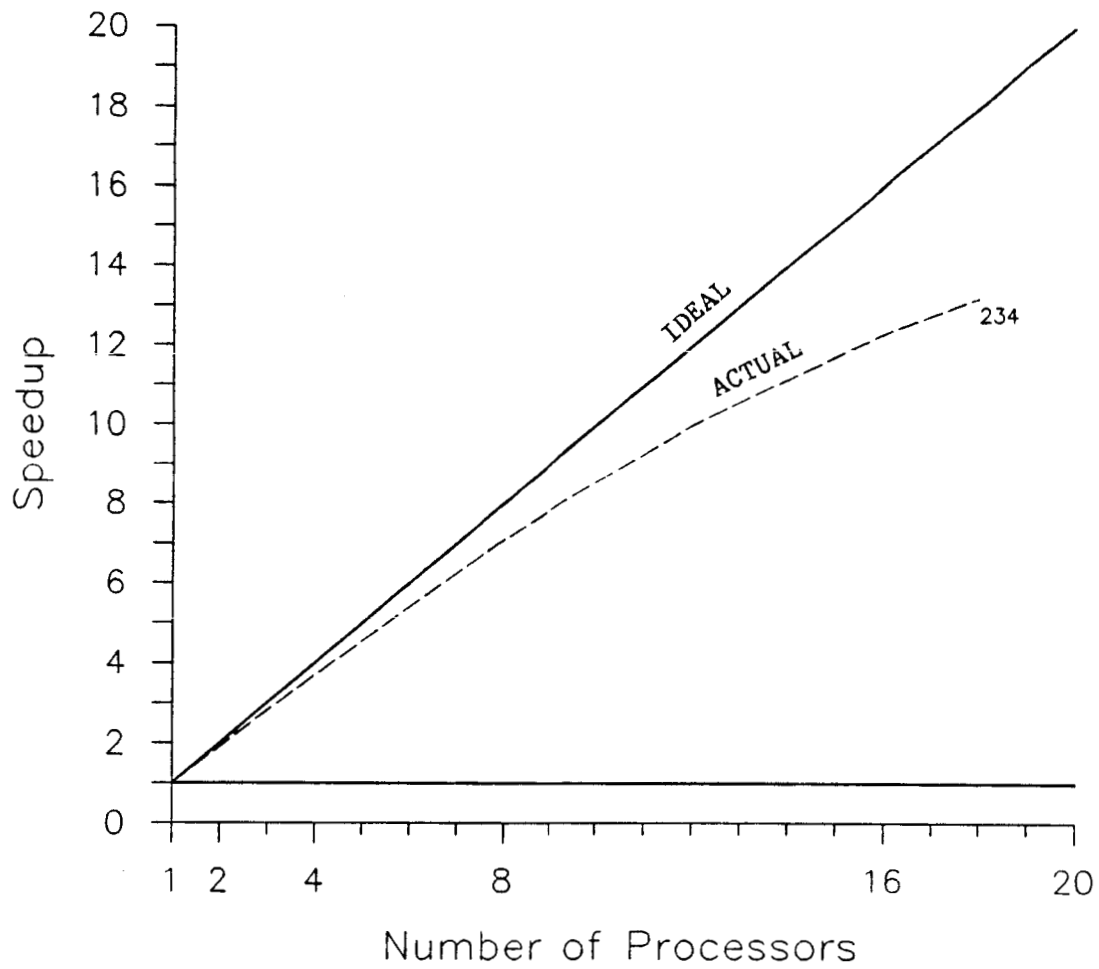


Figure 8. Speedup factor versus number of processors for rectangular panel with hole

ORIGINAL PAGE IS
OF POOR QUALITY

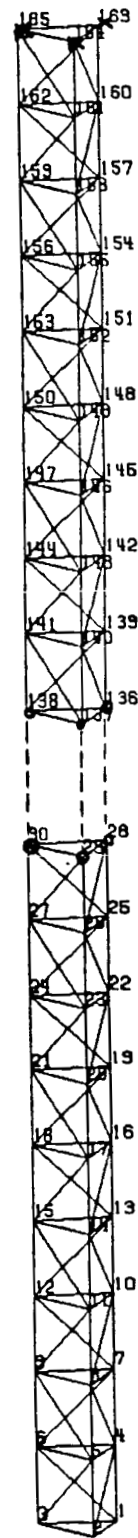
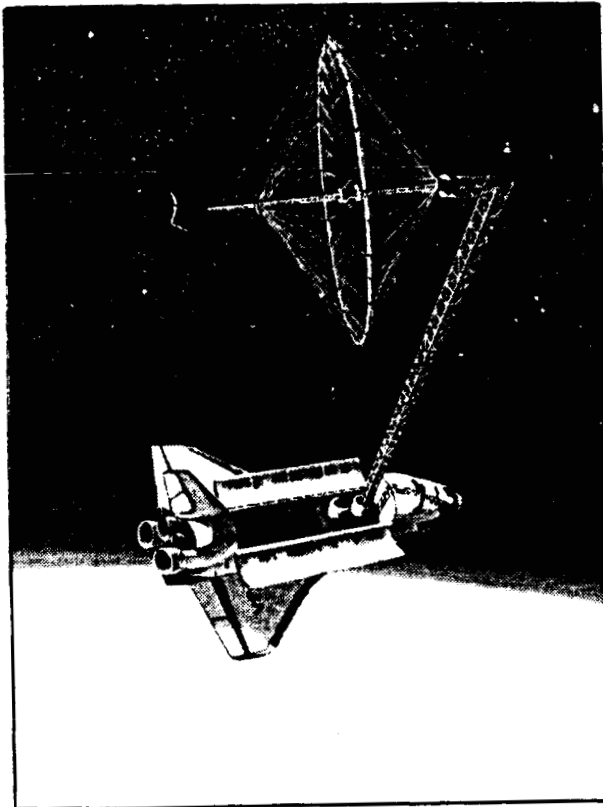


Figure 9. The mast problem

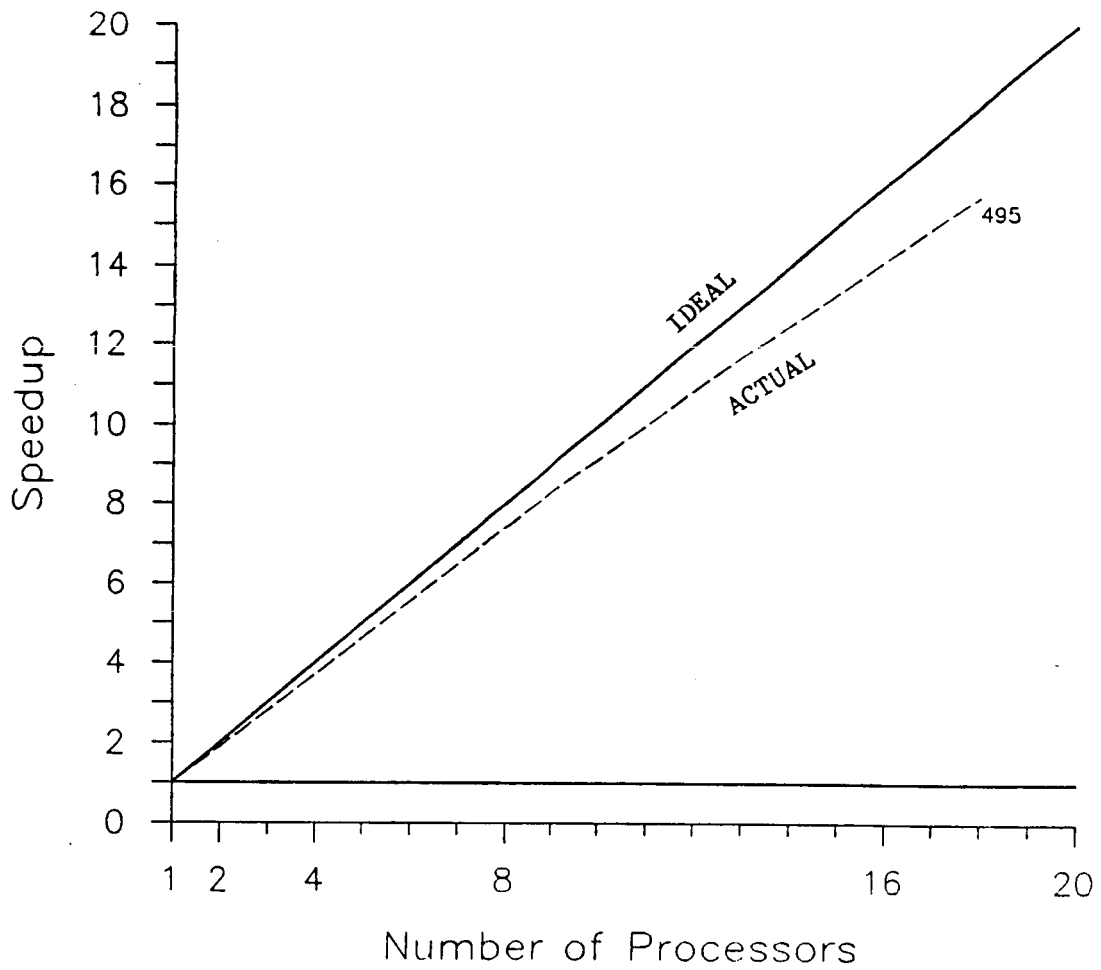


Figure 10. speedup factor versus number of processors for the mast problem

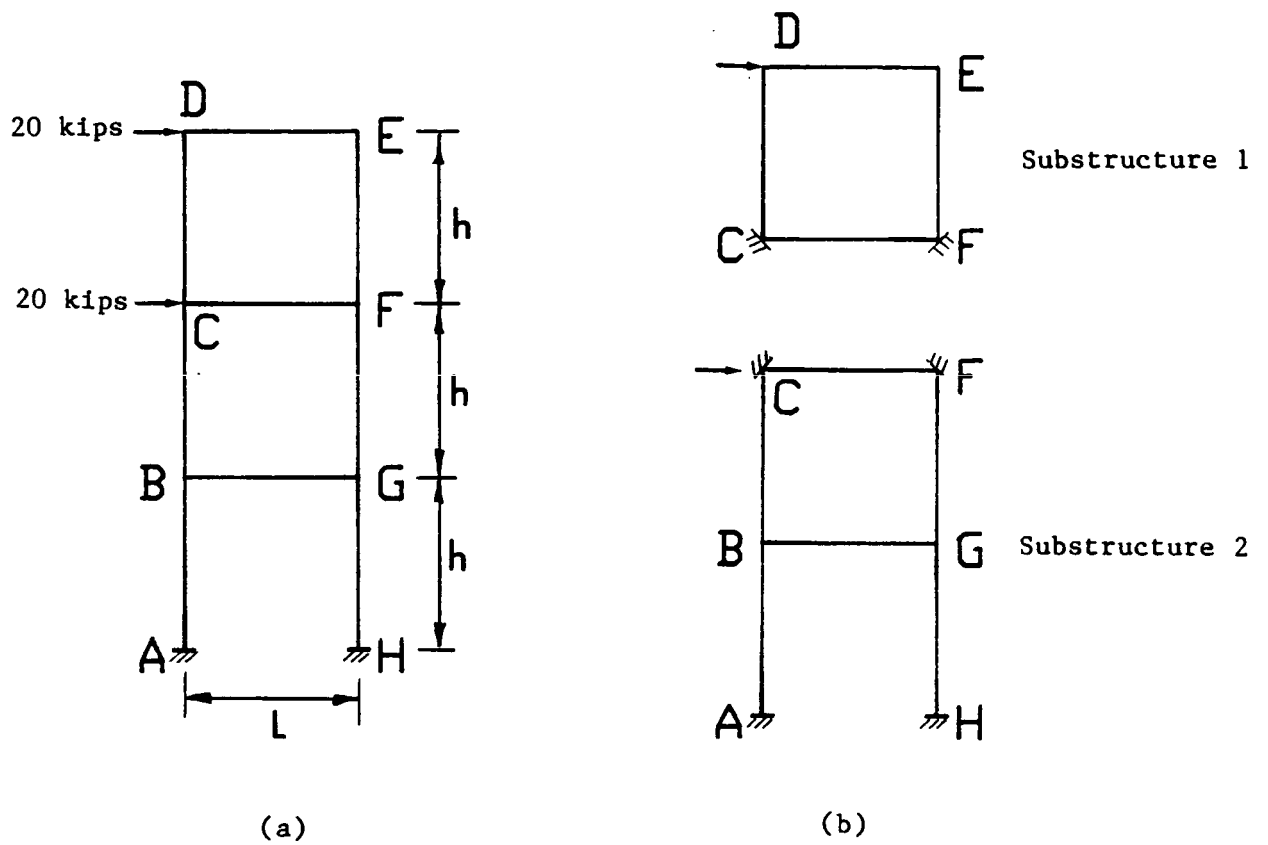


Figure 11. Framed structure with two substructures

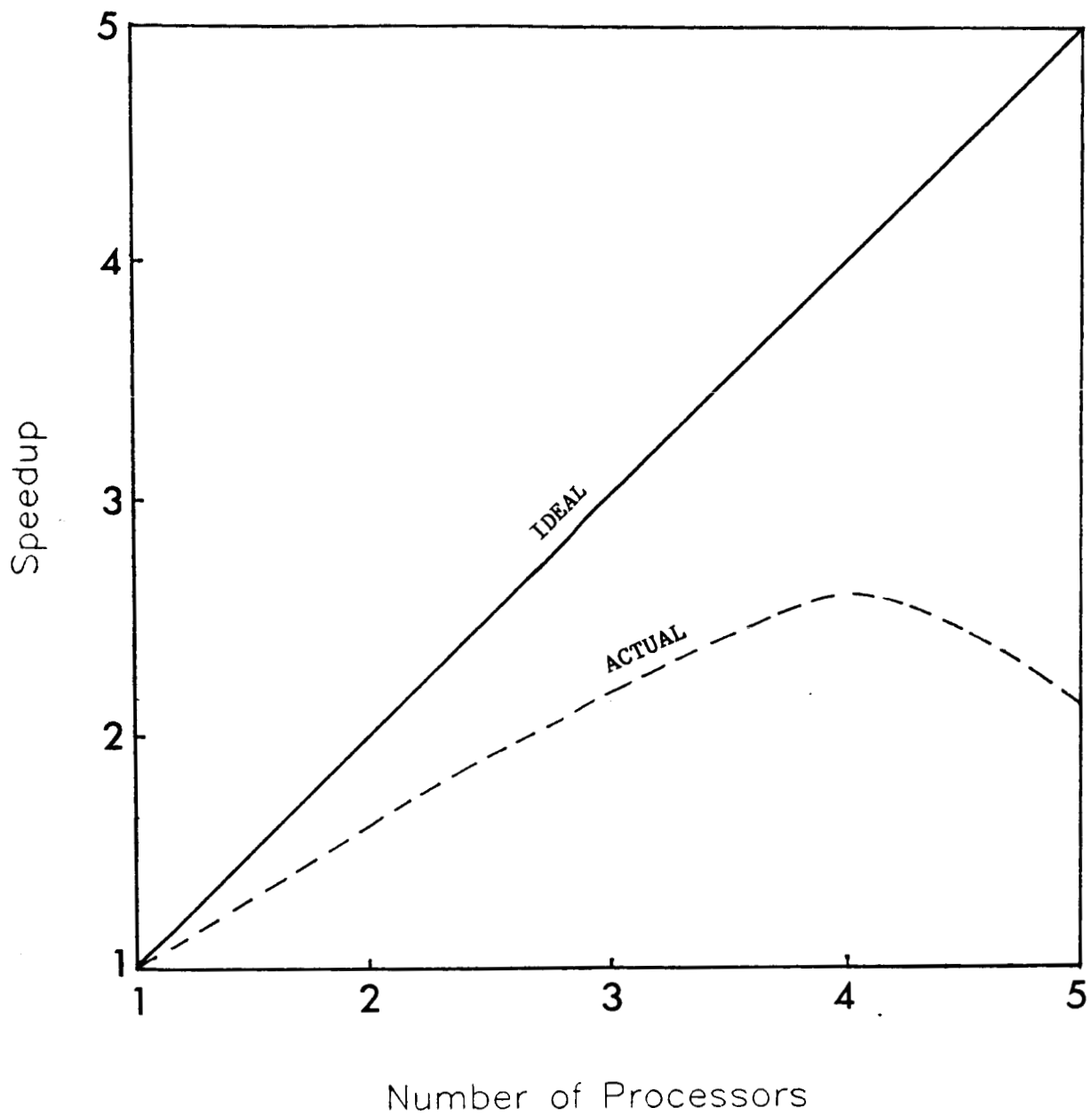


Figure 12. Speedup factor versus the number of processors for 3-story frame using FORCE Subroutine SOLVE

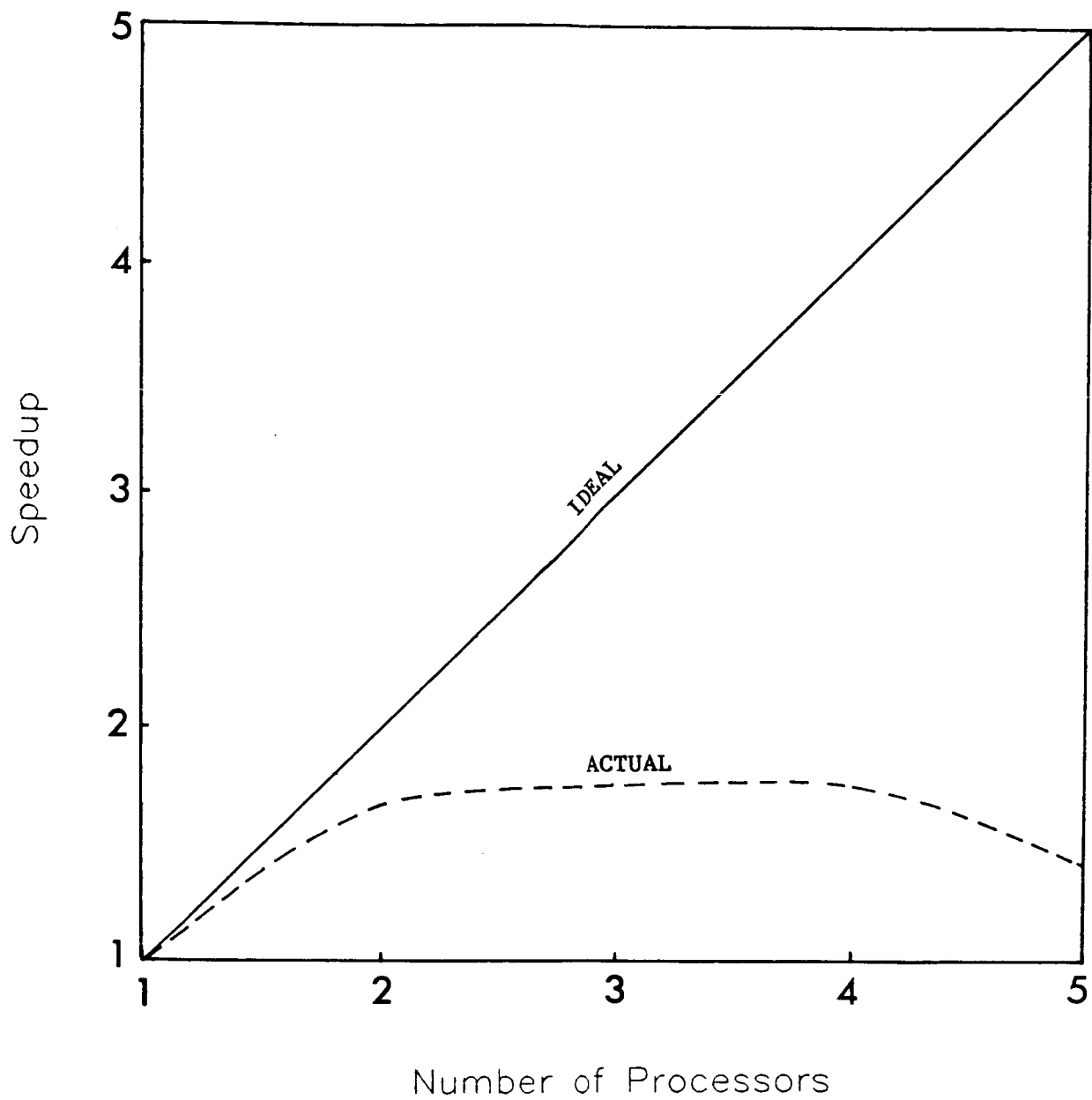


Figure 13. Speedup factor versus the number of processors for 3-story frame with two substructures using FORCE

REFERENCES

1. Barone, T. R., "Beginner's User Manual for Spar," Report No. 78-1, Department of Civil Engineering, Rensselaer Polytechnic Institute, May, 1978.
2. Lotts, C. G., "Introduction to the CSM Testbed: NICE/SPAR," June, 1986.
3. Whetstone, W. D., "SPAR Structural Analysis System Reference Manual," Vol. 1, NASA CR 158970-1, Dec. 1978.
4. Felippa, C.A., "A Command Language for Applied Mechanics Processors," Vols. 1-3 , LMSC-D 78511, Nov. 1983.
5. Jordan, H.F. ,Benten, M.S.,and Ramanan ,A.V., "Force Users Manual, "Computer Systems Design Group, University of Colorado, Jun., 1987.
6. Darbhamulla, S. P., Razzaq, Z., and Storaasli, O. O., "Concurrent Processing in Nonlinear Structural Stability," 27th Structural Dynamics and Materials Conference, AIAA/ASME/ASCE/AHA, San Antonio, Texas, May, 1986.
7. Darbhamulla, S. P., Razzaq, Z., and Storaasli, O. O., "Concurrent Processing for Nonlinear Analysis of Hollow Rectangular Structural Sections," 26th Structural Dynamics and Materials Conference, AIAA/ASME/ASCE/AHA, Orlando, Florida, April, 1985.
8. Darbhamulla, S. P., Razzaq, Z., and Storaasli, O. O., "Concurrent Processing for Nonlinear Analysis of Hollow Rectangular Structural Sections," Engineering with Computers, An International Journal for Computer-aided Mechanical and Structural Engineering, Vol. 2, 1987.
9. Beaufait, F. W., Rowan, W. H., Hoadley, P. G., and Heckett, R. M., Computer Mehods of Strucural Analysis, Printice-Hall, Inc., New Jersey, 1970.

APPENDIX A

RELEVANT SPAR PROCESSORS AND CLAMP PROCEDURES

A.1 Relevant SPAR Processors

A number of SPAR processors are described in Reference 3 some of which are used in the solution procedure presented in section 1.3 of this report. A brief description of the relevant SPAR processors is given in this section. The corresponding page numbers from volume 1 of reference 3 are indicated in the parentheses following the SPAR processor names.

1. Processor TAB (3.1-1)

The function of this processor is to define the finite element model of the structure. The following subprocessors of TAB are utilized:

JLOC defines the joint locations in a structure;

MATC defines the material constants of the structure;

JREF defines the orientation of reference frame associated with the joints;

CON is used in defining constraints and applied loading; and

SA generates a table of shell (or panel) section properties to which reference is made during the definition of SPAR element E41.

All the subprocessors generate their respective data sets which are stored in a data library.

2. Processor ELD (3.2-1)

This processor produces datasets containing element definitions such as the connecting joints, the type of SPAR element used, and the integers pointing to applicable lines in the table of section properties.

3. Processor AUS (5.1-1)

This processor is comprised of an array of subprocessors which are data set constructors. The subprocessors are summarized in tab 5-1 (page 5.1-2,reference3) according to their functional categories. The functional categories include matrix arithmetic and modification of data tables. The following subprocessors of AUS are utilized:

TABLE (2.5.1) creates the data sets which details the number of interface nodes, the number of substructures, and the number of elements in a substructure; and

SYSVEC (5.1.3) is used primarily to represent either the joint displacements and rotations, or the applied forces and moments.

4. Processor E (3.3-1)

This processor generates in a skeletal form an element information packet for each element in the structure, and supplies general information such as the connecting joint numbers, material constants, and section properties data previously identified by processor ELD.

5. Processor EKS (3.4-1)

This processor reads data sets created by processor E and generates the element stiffness matrices.

6. Processor TOPO (4.1-1)

This processor analyzes element interconnection topology and creates data sets (KMAP and AMAP), Reference 4) for the assembly of the system stiffness matrix in SPAR standard sparse-matrix format.

7. Processor K (4.2-1)

This processor assembles unconstrained system stiffness matrix in a SPAR standard sparse-matrix format. The input for K comes from the processors EKS, TOPO and partly from TAB. The output is a data set named K SPAR containing the unconstrained system stiffness matrix

8. Processor INV (4.5-1)

This processor factors assembled system matrix in the SPAR standard sparse-matrix format. It requires the unconstrained system matrix, the constraint definitions, and the topological information in AMAP.

9. Processor SSOL (6.3-1)

This processor computes the displacements and reactions due to applied loading at the joints.

10. Processor GSF (7.1-1)

This processor generates data sets containing element stresses. The input data sets are structural deformations from SSOL. The output data sets are named STRS.

11. Processor PSF (7.2-1)

This processor prints the element stresses and stress resultants from the data sets generated by GSF.

12. Processor VPRT (5.3-1)

This processor is used to edit and display information in SYSVEC format, such as the static displacements and reactions. It is also called the Vector Printer.

A.2 CLAMP Procedures

The substructure analysis using NICE/SPAR requires the use of MACRO subprocessors and CLAMP procedures described below.

1. MACRO Subprocessor

This is a subprocessor within the processor AUS, and is used to define a macrosymbol. The macrosymbols are used to establish variable like items which can perform arithmetic manipulations, access built-in functions, and rename directives.

2. PROCEDURES

The substructure CLAMP procedures END__SUB and GLOB_DAT are used to minimize the user input data. These two procedures are stored in a data file named PRESERVE in the runstream presented in Appendix B. The procedures are described briefly as follows:

Procedure END__SUB scans through the table of contents and stores the sequence number of data sets. Later, the sequence numbers are used for global topology generation. The data sets are then renamed to make them distinct in order to avoid disability of the existing data set.

Procedure GLOB_DAT consists of a number of intermediate procedures such as Inter, Glob__Jloc, Glob__Matc. Procedure Inter searches the interface table for interface joints and separates it. procedure Glob__Jloc translates the substructure joint numbers to global joint numbers, and sets up global joint location data set. Procedure Glob__Matc creates the data set of material constants for the assembled structure.

APPENDIX B
NICE/SPAR RUNSTREAM SAMPLES

The runstreams for the eight-element rectangular panel with hole are presented in this appendix including the outputs. The attached computer listings include:

1. Runstream for eight-element rectangular panel.
2. Output for eight-element rectangular panel.
3. Runstream for rectangular panel with hole.
4. Output for rectangular panel with hole.

A listing of the FORCE program for concurrent solution of simultaneous equations developed by Jordan and his research associates, from the University of Colorado, is also contained in item 3 mentioned above.

RUNSTREAM FOR EIGHT-ELEMENT RECTANGULAR PANEL

```

rmprocl
rm SUB_TABLE
rm SS1.L01
nicespar <<\endinput
*set echo=off
*open 1 ss1.l01 /new
.
.
DUMMY START CARD
[ xqt tab
start 1000
.
*procedure SUB_TABLE
.
. SET UP TABLES
.
.
[ xqt aus

TABLE(NI=3,NJ=1,ITYPE=0): SUBS PARA 1 1
J=1: 2 4 1      . nsubs nnpe iopt
.
. NO. OF ELEMENTS IN SUBSTR.
.
TABLE(NI=2,NJ=1,ITYPE=0): SUBS ELEM 1 1
J=1: 4 4          . no. of elements per sub. (for n subs)

. SUBSTRUCTURE INTERFACES
.
. syntax:
. J=m: j1 s1 j2 s2      m=1,no. of interface nodes
.
.
TABLE(NI=4,NJ=3,ITYPE=0): SUBS SUBI 1 1
J=1: 3 1 1 2
J=2: 6 1 2 2
J=3: 9 1 3 2
.
*end
*call SUB_TABLE
*add preserve
.
.
. DEFINE SUBSTRUCTURE -1
.
[ xqt tab
start 9,4,5,6
matc:1 10000 0.3

```

ORIGINAL PAGE IS
OF POOR QUALITY

```
online=0
jloc: format=1
1 0.0 0.0 0.0 15.0 0.0 0.0 3 1 3
3 0.0 11.5 0.0 15.0 11.5 0.0
sa(1):1 0.1
con=1
zero 3:1,9
[xqt eld
online=0
e41
1 2 5 4 1 2 2
online=1
[xqt topo
[xqt e
[xqt eks
stop
*call end_sub
.
.
. DEFINE SUBSTRUCTURE -2
.
.
[xqt tab
start 9,4,5,6
matc: 1 10000 0.3
jloc: format=1
online=0
1 15. 0. 0. 15. 11.5 0. 3 1
4 22.5 0.0 0.0 30.0 0.0 0.0 2 1 3
2 22.5 11.5 0.0 30.0 11.5 0.0
sa(1):1 0.1
con=1
zero 3:1,9
zero 1:5,9,2
zero 2:9
[xqt eld
online=0
e41: 1 4 6 2
2 6 8 3
4 5 7 6 1 1 2
online=1
[xqt topo
[xqt e
[xqt eks
stop
*call end_sub
*call glob_dat
.
[xqt k
[xqt inv
[xqt aus
alpha
```



```
case titles
1' external load
sysvec
applied forces
case 1
l=1
J=1:2.875
J=4:5.75
J=7:2.875
[xqt ssol
[xqt gsf
[xqt dcu
[xqt psf
[xqt vpri
  tprint stat disp
  tprint stat reac
[xqt exit
endinput
```

**ORIGINAL PAGE IS
OF POOR QUALITY**

ORIGINAL PAGE IS
OF POOR QUALITY

OUTPUT FOR EIGHT-ELEMENT RECTANGULAR PANEL

```

(OL) PUT_message,Comment)
(OL) $root,L0001,C000001)*set echo=off
(DM) OPEN, Ldi: 1, File: SS1.L01 , Attr: new, Block I/O
** BEGIN TAB ** DATA SPACE= 200000 WORDS

1000 JOINTS.
OACTIVE JOINT MOTION COMPONENTS= 1 2 3 4 5 6
OLIB READ ERROR, NU,L,KORE,IERR,IOP= 1 0 196988 -1 11
NAME= MASK BTAB 2 5
O*** ERRORS IN INPUT PREVENT CALCULATION OF QJ(3,3,JT)
EXIT TAB CPUTIME= 7.0 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
TABL COMPLETED.
TABL COMPLETED.
TABL COMPLETED.
EXIT AUS CPUTIME= 5.3 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
MACR COMPLETED.
MACR COMPLETED.
MACR COMPLETED.
EXIT AUS CPUTIME= 1.9 I/O(DIR,BUF)= 0 0

```

```

3 JOINTS.
OACTIVE JOINT MOTION COMPONENTS= 1 2 3
EXIT TAB CPUTIME= 47.6 I/O(DIR,BUF)= 0 0
** BEGIN ELD ** DATA SPACE= 200000 WORDS
EXIT ELD CPUTIME= 7.4 I/O(DIR,BUF)= 0 0
** BEGIN TOPO ** DATA SPACE= 200000 WORDS
(DM) OPEN, Ldi: 25, File: NS.L25 , Attr: scratch, Block I/O
(DM) OPEN, Ldi: 26, File: NS.L26 , Attr: scratch, Block I/O
NO. OF 4-NODE ELEMENTS= 4
OTOTAL NO. OF ELEMENTS= 4
OMAXCON, MAXSUB, ILMAX= 1872 1400 52
OKSIZE,NR5,LR5= 13 1 896
OMAXCON, MAXSUB, ILMAX= 1864 1400 52
OSIZE INDEX= 15, IC1, IC2= 85 33, NR4= 1
(DM) CLOSE, Ldi: 25, File: NS.L25
(DM) CLOSE, Ldi: 26, File: NS.L26
EXIT TOPO CPUTIME= 7.3 I/O(DIR,BUF)= 0 0
** BEGIN E ** DATA SPACE= 200000 WORDS
T= 0.10000E-19-0.10000E-02 0.10000E-04 0.10000E-04
0.20000E+02 0.10000E-03 0.10000E-03 0.10000E-03
ERROR LEVELS= 2 2 0 2 2 2 2

```

```

0
0
L, VOL OR STRUCTURAL NON-STRUCTURAL
TYPE GROUP AREA SUM WEIGHT WEIGHT
E#1 1 0.172500E+03 0.000000E+00 0.000000E+00
0
TOTAL 0.000000E+00 0.000000E+00

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

TOTAL 4-NODES 0.1725000E+03
EXIT E CPUTIME= 6.3 I/O(DIR,BUF)= 0 0
** BEGIN EKS ** DATA SPACE= 200000 WORDS
E41 COMPLETED
EXIT EKS CPUTIME= 2.3 I/O(DIR,BUF)= 0 0
OKAY 1
** BEGIN AUS ** DATA SPACE= 200000 WORDS
MACR COMPLETED.
MACR COMPLETED.
MACR COMPLETED.
MACR COMPLETED.

OKAY 1.2
OKAY 1.4
nsect = 43
nsect[1] = 1
MACR COMPLETED.
EXIT AUS CPUTIME= 9.3 I/O(DIR,BUF)= 0 0
<OL> 10:0 t:515.6 SECT[1] 10.000000
OKAY 2 2 2
OKAY 3 3 3
EXIT TAB CPUTIME= 26.5 I/O(DIR,BUF)= 0 0
** BEGIN ELD ** DATA SPACE= 200000 WORDS
EXIT ELD CPUTIME= 11.9 I/O(DIR,BUF)= 0 0
** BEGIN TOPO ** DATA SPACE= 200000 WORDS
<DM> OPEN, Ldi: 25, File: NS.L25 , Attr: scratch, Block I/O
<DM> OPEN, Ldi: 26, File: NS.L26 , Attr: scratch, Block I/O
NO. OF 4-NODE ELEMENTS= 4
TOTAL NO. OF ELEMENTS= 4
DMAXCON, MAXSUB, ILMAX= 1872 1400 52
OKSIZE,NRE,LRS= 13 1 896
DMAXCON, MAXSUB, ILMAX= 1864 1400 52
OSIZE INDEX= 15, IC1, IC2= 30 32, NR4= 1
<DM> CLOSE, Ldi: 25, File: NS.L25
<DM> CLOSE, Ldi: 26, File: NS.L26
EXIT TOPO CPUTIME= 10.8 I/O(DIR,BUF)= 0 0
** BEGIN E ** DATA SPACE= 200000 WORDS
T= 0.10000E-19-0.10000E-02 0.10000E-04 0.10000E-04
0.20000E+02 0.10000E-03 0.10000E-03 0.10000E-03
ERROR LEVELS= 2 2 0 2 2 2 2 2
0
0
L, VOL OR STRUCTURAL NON-STRUCTURAL
TYPE GROUP AREA SUM WEIGHT WEIGHT
E41 1 0.172500E+03 0.000000E+00 0.000000E+00
0
TOTAL 0.000000E+00 0.000000E+00

```

TOTAL 4-NODES 0.1725000E+03
 EXIT E CPUTIME= 10.7 I/O(DIR,BUF)= 0
 ** BEGIN EKS ** DATA SPACE= 200000 WORDS
 E41 COMPLETED
 EXIT EKS CPUTIME= 4.2 I/O(DIR,BUF)= 0
 OKAY 1
 ** BEGIN AUS ** DATA SPACE= 200000 WORDS
 MACR COMPLETED.
 MACR COMPLETED.
 MACR COMPLETED.
 MACR COMPLETED.
 OKAY 1.2
 OKAY 1.4
 nsect = 43
 nsect[2] = 1

ORIGINAL PAGE IS
 OF POOR QUALITY

MACR COMPLETED.
 EXIT AUS CPUTIME= 12.1 I/O(DIR,BUF)= 0
 <CL> 10:0 t:I SECTL[1] 10.000000
 ** BEGIN AUS ** DATA SPACE= 200000 WORDS
 MACR COMPLETED.
 MACR COMPLETED.
 MACR COMPLETED.
 MACR COMPLETED.
 <CL> 10:0 t:I GJNT[1] 1
 <CL> 10:0 t:I GJNT[2] 2
 <CL> 10:0 t:I GJNT[3] 3
 <CL> 10:0 t:I GJNT[4] 4
 <CL> 10:0 t:I GJNT[5] 5
 <CL> 10:0 t:I GJNT[6] 6
 <CL> 10:0 t:I GJNT[7] 7
 <CL> 10:0 t:I GJNT[8] 8
 <CL> 10:0 t:I GJNT[9] 9
 <CL> 10:0 t:I GJNT[10] 8
 <CL> 10:0 t:I GJNT[11] 6
 <CL> 10:0 t:I GJNT[12] 9
 <CL> 10:0 t:I GJNT[13] 10
 <CL> 10:0 t:I GJNT[14] 11
 <CL> 10:0 t:I GJNT[15] 12
 <CL> 10:0 t:I GJNT[16] 13
 <CL> 10:0 t:I GJNT[17] 14
 <CL> 10:0 t:I GJNT[18] 15
 EXIT AUS CPUTIME= 130.1 I/O(DIR,BUF)= 0
 ** BEGIN AUS ** DATA SPACE= 200000 WORDS
 DEFI COMPLETED.
 DEFI COMPLETED.
 TABL COMPLETED.
 MACR COMPLETED.
 TABL COMPLETED.
 TABL COMPLETED.
 TABL COMPLETED.
 TABL COMPLETED.
 TABL COMPLETED.
 TABL COMPLETED.
 TABL COMPLETED.
 TABL COMPLETED.

ORIGINAL PAGE IS
OF POOR QUALITY

```

EXIT AUS CPUTIME= 59.3 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
DEFI COMPLETED.
DEFI COMPLETED.
TABL COMPLETED.

EXIT AUS CPUTIME= 19.6 I/O(DIR,BUF)= 0 0
START 15 4,5,6
EXIT TAB CPUTIME= 11.6 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
MACR COMPLETED.
# of elements in substructure 1 - 4
NJ of DEF dataset is 56
MACR COMPLETED.
# of elements of substructure 2 - 4
MACR COMPLETED.
TABL COMPLETED.
MACR COMPLETED.
TABL COMPLETED.
MACR COMPLETED.
TABL COMPLETED.
MACR COMPLETED.
TABL COMPLETED.

EXIT AUS CPUTIME= 69.3 I/O(DIR,BUF)= 0 0
** BEGIN ELD ** DATA SPACE= 200000 WORDS
EXIT ELD CPUTIME= 16.8 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
TABL COMPLETED.

EXIT AUS CPUTIME= 4.0 I/O(DIR,BUF)= 0 0
** BEGIN TOPO ** DATA SPACE= 200000 WORDS
<DM> OPEN, Ldi: 25, File: NS.L25 , Attr: scratch, Block I/O
<DM> OPEN, Ldi: 26, File: NS.L26 , Attr: scratch, Block I/O
NO. OF 4-NODE ELEMENTS= 8
OTOTAL NO. OF ELEMENTS= 8
OMAXCON, MAXSUB, ILMAX= 1872 1400 52
OKSIZE,NR5,LR5= 18 1 896
OMAXCON, MAXSUB, ILMAX= 1864 1400 52
OSIZE INDEX= 23, IC1, IC2= 221 70, NR4= 1
<DM> CLOSE, Ldi: 25, File: NS.L25
<DM> CLOSE, Ldi: 26, File: NS.L26
EXIT TOPO CPUTIME= 18.0 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
DEFI COMPLETED.
DEFI COMPLETED.
UNIO COMPLETED.

EXIT AUS CPUTIME= 7.9 I/O(DIR,BUF)= 0 0
** BEGIN K ** DATA SPACE= 200000 WORDS
EXIT K CPUTIME= 10.9 I/O(DIR,BUF)= 0 0
** BEGIN INV ** DATA SPACE= 200000 WORDS
ONSING,NNEG= 0 0
EXIT INV CPUTIME= 6.6 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
ALPH COMPLETED.
SYSV COMPLETED.

```

EXIT AUS CPUTIME= 4.8 I/O(DIR,BUF)= 0 0
 ** BEGIN SSOL ** DATA SPACE= 200000 WORDS
 <DM> OPEN, Ldi: 26, File: NS.L26, Attr: scratch, Block I/O

ORIGINAL PAGE IS
 OF POOR QUALITY

0 CASE F*U U*KU ERR

1 0.3450000E+00 0.3450001E+00 -0.8638353E-07
 <DM> CLOSE, Ldi: 26, File: NS.L26
 EXIT SSOL CPUTIME= 16.8 I/O(DIR,BUF)= 0 0
 ** BEGIN DCU ** DATA SPACE= 200000 WORDS

0

EXIT DCU CPUTIME= 0.2 I/O(DIR,BUF)= 0 0
 ** BEGIN VPRT ** DATA SPACE= 200000 WORDS

1STATIC DISPLACEMENTS.

ID= 1/ 1/ 1

external load

JOINT	1	2	3
1	0.300E-01	-0.345E-02	0.000E+00*
2	0.225E-01	-0.345E-02	0.000E+00*
3	0.150E-01	-0.345E-02	0.000E+00*
4	0.300E-01	-0.173E-02	0.000E+00*
5	0.225E-01	-0.173E-02	0.000E+00*
6	0.150E-01	-0.173E-02	0.000E+00*
7	0.300E-01	-0.759E-07	0.000E+00*
8	0.225E-01	-0.521E-07	0.000E+00*
9	0.150E-01	-0.296E-07	0.000E+00*
10	0.750E-02	-0.345E-02	0.000E+00*
11	0.000E+00*	-0.345E-02	0.000E+00*
12	0.750E-02	-0.173E-02	0.000E+00*
13	0.000E+00*	-0.173E-02	0.000E+00*
14	0.750E-02	-0.120E-07	0.000E+00*
15	0.000E+00*	0.000E+00*	0.000E+00*

1STATIC REACTIONS, FORCE ERRORS.

ID= 1/ 1/ 1

external load

JOINT	1	2	3
1	-0.119E-05	-0.283E-06	0.000E+00*
2	0.626E-06	0.119E-06	0.000E+00*
3	-0.121E-05	0.249E-06	0.000E+00*
4	0.143E-05	0.586E-07	0.000E+00*
5	0.400E-06	-0.149E-06	0.000E+00*
6	-0.341E-07	-0.108E-07	0.000E+00*
7	-0.477E-06	0.539E-06	0.000E+00*
8	-0.212E-06	-0.394E-06	0.000E+00*
9	-0.999E-07	-0.660E-07	0.000E+00*
10	0.597E-06	-0.395E-06	0.000E+00*
11	-0.288E+01*	-0.684E-07	0.000E+00*
12	0.569E-08	0.290E-06	0.000E+00*
13	-0.575E+01*	0.167E-06	0.000E+00*
14	-0.293E-07	0.671E-07	0.000E+00*
15	-0.287E+01*	0.178E-05*	0.000E+00*

EXIT VPRT CPUTIME= 4.2 I/O(DIR,BUF)= 0 0

<DM> CLOSE, Ldi: 1, File: SS1.L01

<CL> CSS exhausted

ENDRUN called by CLIP

RUNSTREAM FOR RECTANGULAR PANEL WITH HOLE

```
rm PAN1.L01
rmproc1
nicespar <<\endinput
*set echo=off
*open 1 pan1.l01
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
.
DUMMY START CARD
[<qt tab
start 1000
```

```
*procedure SUB_TABLE
```

```
. SET UP TABLES
```

```
[<qt aus
```

```
TABLE(NI=3,NJ=1,ITYPE=0): SUBS PARA 1 1
```

```
J=1: 2 4 1 . nsubs nnpa iopt
```

```
. NO. OF ELEMENTS IN SUBSTR.
```

```
TABLE(NI=2,NJ=1,ITYPE=0): SUBS ELEM 1 1
```

```
J=1: 16 44 . no. of elements per sub. (for n subs)
```

```
. SUBSTRUCTURE INTERFACES
```

```
. syntax:
```

```
. J=m: j1 s1 j2 s2 m=1,no. of interface nodes
```

```
TABLE(NI=4,NJ=8,ITYPE=0): SUBS SUBI 1 1
```

```
J=1: 17 1 1 2
```

```
J=2: 18 1 2 2
```

```
J=3: 19 1 3 2
```

```
J=4: 20 1 4 2
```

```
J=5: 21 1 5 2
```

```
J=6: 22 1 6 2
```

```
J=7: 23 1 7 2
```

```
J=8: 24 1 8 2
```

```
*end
```

```
*call SUB_TABLE
```

```
*add preserve
```

```
. DEFINE SUBSTRUCTURE -1
```

```
[<qt tap
```

```
start 24,4,5,6
```

```
match:1 10000 0.3
```

```
online=0
```

ORIGINAL PAGE IS
OF POOR QUALITY

jloc: format=1

1	0.160000E+02	0.575000E+01	0.000000E+00
2	0.157071E+02	0.645711E+01	0.000000E+00
3	0.150000E+02	0.675000E+01	0.000000E+00
4	0.142929E+02	0.645711E+01	0.000000E+00
5	0.140000E+02	0.575000E+01	0.000000E+00
6	0.142929E+02	0.704289E+01	0.000000E+00
7	0.150000E+02	0.475000E+01	0.000000E+00
8	0.157071E+02	0.504289E+01	0.000000E+00
9	0.164444E+02	0.575000E+01	0.000000E+00
10	0.161516E+02	0.690155E+01	0.000000E+00
11	0.150000E+02	0.719444E+01	0.000000E+00
12	0.138484E+02	0.690155E+01	0.000000E+00
13	0.135556E+02	0.575000E+01	0.000000E+00
14	0.138484E+02	0.459845E+01	0.000000E+00
15	0.150000E+02	0.430556E+01	0.000000E+00
16	0.161516E+02	0.459845E+01	0.000000E+00
17	0.170000E+02	0.575000E+01	0.000000E+00
18	0.170000E+02	0.775000E+01	0.000000E+00
19	0.150000E+02	0.775000E+01	0.000000E+00
20	0.130000E+02	0.775000E+01	0.000000E+00
21	0.130000E+02	0.575000E+01	0.000000E+00
22	0.130000E+02	0.375000E+01	0.000000E+00
23	0.150000E+02	0.375000E+01	0.000000E+00
24	0.170000E+02	0.375000E+01	0.000000E+00

ss(1):1 0.1

online=0

con=1

zero 3:1,24

[xat eld

reset LREC=1230

E41

1	9	10	2 #
2	10	11	3 #
3	11	12	4 #
4	12	13	5 #
5	14	13	5 #
7	15	14	6 #
8	16	15	7 #
1	9	16	8 #
9	17	18	10 #
10	18	19	11 #
11	19	20	12 #
12	20	21	13 #
14	22	21	13 #
15	23	22	14 #
16	24	23	15 #
9	17	24	16 #

[xat topo

[xat e

[xat eks

stop

rcall end_sub

. DEFINE SUBSTRUCTURE -2-

ORIGINAL PAGE IS
OF POOR QUALITY

[not tab

start 62,4,5,6

matc: 1 10000 0.3

plcc: format=1

1	0.170000E+02	0.575000E+01	0.000000E+00
2	0.170000E+02	0.775000E+01	0.000000E+00
3	0.150000E+02	0.775000E+01	0.000000E+00
4	0.130000E+02	0.775000E+01	0.000000E+00
5	0.130000E+02	0.575000E+01	0.000000E+00
6	0.130000E+02	0.375000E+01	0.000000E+00
7	0.130000E+02	0.375000E+01	0.000000E+00
8	0.170000E+02	0.375000E+01	0.000000E+00
9	0.000000E+00	0.000000E+00	0.000000E+00
10	0.433333E+01	0.000000E+00	0.000000E+00
11	0.866667E+01	0.000000E+00	0.000000E+00
12	0.130000E+02	0.000000E+00	0.000000E+00
13	0.150000E+02	0.000000E+00	0.000000E+00
14	0.170000E+02	0.000000E+00	0.000000E+00
15	0.213333E+02	0.000000E+00	0.000000E+00
16	0.256667E+02	0.000000E+00	0.000000E+00
17	0.300000E+02	0.000000E+00	0.000000E+00
18	0.000000E+00	0.125000E+01	0.000000E+00
19	0.433333E+01	0.125000E+01	0.000000E+00
20	0.866667E+01	0.125000E+01	0.000000E+00
21	0.130000E+02	0.125000E+01	0.000000E+00
22	0.150000E+02	0.125000E+01	0.000000E+00
23	0.170000E+02	0.125000E+01	0.000000E+00
24	0.213333E+02	0.125000E+01	0.000000E+00
25	0.256667E+02	0.125000E+01	0.000000E+00
26	0.300000E+02	0.125000E+01	0.000000E+00
27	0.000000E+00	0.375000E+01	0.000000E+00
28	0.433333E+01	0.375000E+01	0.000000E+00
29	0.866667E+01	0.375000E+01	0.000000E+00
30	0.213333E+02	0.375000E+01	0.000000E+00
31	0.256667E+02	0.375000E+01	0.000000E+00
32	0.300000E+02	0.375000E+01	0.000000E+00
33	0.000000E+00	0.575000E+01	0.000000E+00
34	0.433333E+01	0.575000E+01	0.000000E+00
35	0.866667E+01	0.575000E+01	0.000000E+00
36	0.213333E+02	0.575000E+01	0.000000E+00
37	0.256667E+02	0.575000E+01	0.000000E+00
38	0.300000E+02	0.575000E+01	0.000000E+00
39	0.000000E+00	0.775000E+01	0.000000E+00
40	0.433333E+01	0.775000E+01	0.000000E+00
41	0.866667E+01	0.775000E+01	0.000000E+00
42	0.213333E+02	0.775000E+01	0.000000E+00
43	0.256667E+02	0.775000E+01	0.000000E+00
44	0.300000E+02	0.775000E+01	0.000000E+00
45	0.000000E+00	0.102500E+02	0.000000E+00

ORIGINAL PAGE IS
OF POOR QUALITY

46	0.433333E+01	0.102500E+02	0.000000E+00
47	0.866667E+01	0.102500E+02	0.000000E+00
48	0.130000E+02	0.102500E+02	0.000000E+00
49	0.150000E+02	0.102500E+02	0.000000E+00
50	0.170000E+02	0.102500E+02	0.000000E+00
51	0.213333E+02	0.102500E+02	0.000000E+00
52	0.256667E+02	0.102500E+02	0.000000E+00
53	0.300000E+02	0.102500E+02	0.000000E+00
54	0.000000E+00	0.115000E+02	0.000000E+00
55	0.433333E+01	0.115000E+02	0.000000E+00
56	0.866667E+01	0.115000E+02	0.000000E+00
57	0.130000E+02	0.115000E+02	0.000000E+00
58	0.150000E+02	0.115000E+02	0.000000E+00
59	0.170000E+02	0.115000E+02	0.000000E+00
60	0.213333E+02	0.115000E+02	0.000000E+00
61	0.256667E+02	0.115000E+02	0.000000E+00
62	0.300000E+02	0.115000E+02	0.000000E+00

sa(1):1 0.1

con=1

ZERO 3:1,62

ZERO 1:17,26,9

ZERO 1:32,44,6

ZERO 1:53,62,9

ZERO 2:62

(xat elc

e41

9 10 19 18 1 8 1

18 19 28 27 1 2 1

20 21 6 29

21 22 7 6 1 2 1

23 24 30 3

24 25 31 30 1 2 1

27 28 34 35 1 2 1

29 6 5 35

8 30 36 1

30 31 37 36 1 2 1

33 34 40 39 1 2 1

35 5 4 41

1 36 42 2

36 37 43 42 1 2 1

39 40 46 45 1 2 1

41 4 48 47

4 3 49 43

3 2 50 49

2 42 51 50

42 43 52 51 1 2 1

45 46 55 54 1 8 1

online=1

(xat tops

(xat e

(xat eks

stop

ORIGINAL PAGE IS
OF POOR QUALITY

```
*call end_sub
*call glob_dat
,
[xt k
[xt inv
[xt sus
sysvec
applied forces
case 1
  i=i
  J= 25: 0.625
  J= 34 : 1.875
  J= 43 : 2.250
  J= 49 : 2.0
  J= 55 : 2.250
  J= 61 : 1.875
  J= 70 : 0.625
[xt ssol
[xt scu
[xt vprt
  tprint stat disp
  tprint stat read
[xt exit
endinput
```

ORIGINAL PAGE IS
OF POOR QUALITY

C !!! link with /usr/u1/elp/lib/nslib.a !!!
 Force LUD of NP ident ME
 C.....Test program for parallel PLU decomposition using Force versions
 C.....of "Linpack" routines.
 C.....This program calls a (single stream) routine to read in a matrix
 C.....and vector, then SGEFA is called to factor the matrix into a PLU form,
 C.....then SGESL is called to solve the system. The computation is timed,
 C.....and then a (single stream) routine is called to output the results.

Shared INTEGER N, IPVT(500), INFO
 Shared DOUBLE PRECISION A(500,500) ,B(500)
 shared INTEGER ITM1, ITM2, ITM3
 shared REAL DELTA1, DELTA2, RATE1, RATE2
 Private integer amax,numjnt,dof,ierr,rmax,bw
 Private integer rowptr(1000), rowlen(1000)
 Private double precision amat(50000),x(1000)
 Private integer i, j
 End declarations

Barrier
 amax = 10000
 rmax = 1000
 call getrow(amat,amax,rowptr,rowlen,rmax,
 : '/usr/u1/venkatesh/panel/MAS.L01',
 : x,b,numjnt,dof,bw,ierr)
 if (ierr.ne.0) write(*,5) ierr
 5 format('error return code = ',i2)
 N = numjnt*dof
 print*, 'starts to print matrix'
 do 1 i = 1,N
 write(*,2) i,rowptr(i),rowlen(i),amat(rowptr(i))
 1 continue
 2 format('row ',i3,' start ',i6,'lth=',i3,'diag= ',g15.5)
 C.....Code to transfer from vector to symetric array, a
 print*, 'starts symmetric array'
 do 20 irow = 1,N
 ip = rowptr(irow)
 C ilen = rowlen(irow)
 ilen = (irow-1) + rowlen(irow)
 do 10 icol = irow,ilen
 a(irow,icol) = amat(ip)
 ip = ip + 1
 10 continue
 C.....zero out the remainder of the array
 do 15 icol = ilen+1, N
 a(irow,icol) = 0.
 15 continue
 20 continue
 print*, 'o k a y 3'
 do 40 icol = 1, N
 do 30 irow = icol, N
 a(irow,icol) = a(icol,irow)
 30 continue
 40 continue

```

C      CALL PRNTB2(A, B, N)
      CALL CFrtic(ITM1)
      End barrier

      Forcecall SGEFA(A, 500, N, IPVT, INFO)
      IF (INFO .NE. 0) THEN
C      matrix is singular...
        if (me .eq. 1) print *, 'matrix is singular'
        Join
      END IF
      print*, 'o k a y 4'

      IF (ME .EQ. 1) CALL CFrtic(ITM2)

      Forcecall SGESL(A, 500, N, IPVT, B, 0)

      Barrier
      print*, 'o k a y 5'
      CALL CFrtic(ITM3)
      CALL PRINTB(A, B, N)

      DELTA1 = (ITM2 - ITM1)/50.0
      DELTA2 = (ITM3 - ITM2)/50.0
      OPEN(8, FILE="/usr/u1/venkatesh/time.dat", CPU=1)
      IF (DELTA1 .NE. 0) THEN
        RATE1 = (2.0*N*N*N/3.0)*1.0E-6/DELTA1
        WRITE (8, 400) NP, N, ME, DELTA1, RATE1
      ELSE
        WRITE (8, 400) NP, N, ME, DELTA1, 0.0
        WRITE (8, 402)
      ENDIF
      IF (DELTA2 .NE. 0) THEN
        RATE2 = (N*N)*1.0E-6/DELTA2
        WRITE (8, 401) ME, DELTA2, RATE2
      ELSE
        WRITE (8, 401) ME, DELTA2, 0.0
        WRITE (8, 402)
      ENDIF
      print*, 'o k a y 6'
400      FORMAT('  Number of processes =', I3, ' Matrix order =', I4, /
+      ' matrix decomposition: ', F12.4, ' seconds (' , F5.3, ' MFLOPS)')
401      FORMAT(
+      ' Fwd/back subst:      ', F12.4, ' seconds (' , F5.3, ' MFLOPS)')
402      FORMAT(' No elapsed time measured.')
      End barrier
      Join
      END

```

```

Forcesub SGESL(A, LDA, N, IPVT, B, JOB) of NP ident me
C  SGESL: Force parallel version of LINPACK's SGESL which solves
C      Ax=b, using (column orientated) forward/backward subst.
C      A call to SGEFA must precede the call to SGESL.
C      SGEFA provides the PLU factorization of A.
C  A:      the working matrix
C  LDA:    the declared dimension of A
C  N:      the working dimension of A
C  IPVT:   the pivot array
C  B:      the input/output vector (solved in place)
C          B is the only argument altered by SGESL.
C  JOB:    job selector:  job=0: solves Ax=b
C                      job=1: solves Transpose(A)x=b, not implemented
C          All argument MUST be declared "Shared" in the calling module!
integer LDA, N, IPVT(LDA), JOB
double precision A(LDA,LDA), B(LDA)
Private integer I, K, L
Shared real      TEM
End declarations

C  first solve: L*Y = B
do 20 K = 1, N-1
  Barrier
  L = IPVT(K)
  TEM = B(L)
  B(L) = B(K)
  B(K) = TEM
  End barrier
  Presched do 22 I = K+1, N
    B(I) = B(I) + TEM * A(I,K)
22  End presched do
20  continue

C  then solve: U*X = Y
do 40 K = N, 1, -1
  Barrier
  B(K) = B(K)/A(K,K)
  TEM = -B(K)
  End barrier
  Presched do 42 I = 1, K-1
    B(I) = B(I) + TEM * A(I,K)
42  End presched do
40  continue
  Barrier
  End barrier
  RETURN
END

```

```

Forcesub SGEFA(A, LDA, N, IPVT, INFO) of NP ident ME
INTEGER LDA, N, IPVT(LDA), INFO
DOUBLE PRECISION A(LDA, LDA)

```

C

```

Private REAL ABSMAX
Private REAL T
Private INTEGER KK, I, MAXI, J, KP1
Shared INTEGER IALL
Shared REAL PIV, ALLMAX
Shared logical ILOCK
End declarations

```

```

Barrier
INFO = 0
ALLMAX = 0.0
End barrier

```

C.....For each row of the matrix

```

DO 1000 KK = 1, N-1

IF(INFO.NE.0) GOTO 2000

```

C.....Reset local maxima

```

ABSMAX = 0.0

```

C.....Find the pivot row

```

Presched DO 100 I = KK, N
T = ABS( A(I, KK) )
IF (ABSMAX .LT. T) THEN
MAXI = I
ABSMAX = T
ENDIF
100 End Presched DO

```

C.....Update the Shared Information if necessary

```

Critical ILOCK
IF (ALLMAX .LT. ABSMAX) THEN
IALL = MAXI
ALLMAX = ABSMAX
ENDIF
End critical

```

```

Barrier
IF (ALLMAX .EQ. 0.0) THEN
INFO = KK
IPVT(KK)=KK
ELSE
IPVT(KK)=IALL
ENDIF
End Barrier

```

C.....If the matrix is singular then pass the information

```
IF (INFO .EQ. KK) GOTO 1000
MAXI = IALL
IF (MAXI .NE. KK) THEN
```

C.....Swap rows if necessary

```
Presched DO 110 J = KK, N
    TEMP = A(MAXI, J)
    A(MAXI, J) = A(KK, J)
    A(KK, J) = TEMP
110 End Presched DO
ENDIF
```

C.....Self schedule row reductions

```
KP1 = KK + 1
Barrier
    PIV = -1.0/A(KK, KK)
End Barrier

Selfsched DO 130 I = KP1, N
    TEMP = PIV*A(I, KK)
    A(I, KK) = TEMP
    DO 120 J = KP1, N
        A(I, J) = A(I, J) + TEMP*A(KK, J)
120 CONTINUE
130 End Selfsched DO

Barrier
    ALLMAX=0
End barrier
```

1000 CONTINUE

2000 RETURN
END

***** i/o routines (replacable)*****

```
subroutine printb(a, b, n)
C.....this routine prints B(N) & A(N,N)
C.....where B(N) is the solution vector, and A(N,N) is the working matrix.
C.....the user may replace this single stream subroutine.
double precision b(1), a(500,500)
open(8,FILE="/usr/u1/venkatesh/panel/venk.dat",CPU=1)
rewind (8)
do 10 i=1,n
10 write(8,90) 'b(',i,')=',b(i)
close (8)
90 format(a,i3,a,e14.7)
return
end
```



```

      subroutine prntb2(a, b, n)
C.....this routine prints B(N) & A(N,N)
C.....where B(N) is the solution vector, and A(N,N) is the working matrix.
C.....the user may replace this single stream subroutine.
      double precision b(1), a(500,500)
      open(8,FILE="/usr/u1/venkatesh/panel/venk.predat",CPU=1)
      rewind (8)
      do 10 i=1,n
10      write(8,90) 'b(',i,')=' ,b(i),(a(i,j), j=1,n)
      close (8)
90      format(a,i3,a,300e14.7)
      return
      end

```

OUTPUT FOR RECTANGULAR PANEL WITH HOLE

```

<CL> PUT_message,Commnt>
<CL> $root,L0001,C00001)*set echo=off
<DM> OPEN, Ldi: 1, File: PAN1.L01 , Attr: new, Block I/O
** BEGIN TAB ** DATA SPACE= 200000 WORDS

1000 JOINTS.
OACTIVE JOINT MOTION COMPONENTS= 1 2 3 4 5 6
OLIB READ ERROR, NU,L,KORE,iERR,IOP= 1 0 198988 -1 11
NAME= MASK BTAB 2 5
0*** ERRORS IN INPUT PREVENT CALCULATION OF QJ(3,3,JT)
EXIT TAB CPUTIME= 7.5 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
TABL COMPLETED.
TABL COMPLETED.
TABL COMPLETED.
EXIT AUS CPUTIME= 5.7 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
MACR COMPLETED.
MACR COMPLETED.
MACR COMPLETED.
EXIT AUS CPUTIME= 1.8 I/O(DIR,BUF)= 0 0

24 JOINTS.
OACTIVE JOINT MOTION COMPONENTS= 1 2 3
EXIT TAB CPUTIME= 50.0 I/O(DIR,BUF)= 0 0
LREC= 1280
** BEGIN ELD ** DATA SPACE= 200000 WORDS
EXIT ELD CPUTIME= 8.8 I/O(DIR,BUF)= 0 0
** BEGIN TOPO ** DATA SPACE= 200000 WORDS
<DM> OPEN, Ldi: 25, File: NS.L25 , Attr: scratch, Block I/O
<DM> OPEN, Ldi: 26, File: NS.L26 , Attr: scratch, Block I/O
NO. OF 4-NODE ELEMENTS= 16
OTOTAL NO. OF ELEMENTS= 16
OMAXCON, MAXSUB, ILMAX= 1872 1400 52
OKSIZE,NR5,LR5= 28 1 896
OMAXCON, MAXSUB, ILMAX= 1864 1400 52
OSIZE INDEX= 66, IC1, IC2= 967 191, NR4= 1
<DM> CLOSE, Ldi: 25, File: NS.L25
<DM> CLOSE, Ldi: 26, File: NS.L26
EXIT TOPO CPUTIME= 7.2 I/O(DIR,BUF)= 0 0
** BEGIN E ** DATA SPACE= 200000 WORDS
T= 0.10000E-19-0.10000E-02 0.10000E-04 0.10000E-04
0.20000E+02 0.10000E-03 0.10000E-03 0.10000E-03
ERROR LEVELS= 2 2 0 2 2 2 2
0
0
L, VOL OR STRUCTURAL NON-STRUCTURAL
TYPE GROUP AREA SUM WEIGHT WEIGHT
E41 1 0.131716E+02 0.000000E+00 0.000000E+00

```

```

0
TOTAL                0.000000E+00 0.000000E+00

TOTAL 4-NODE@ 0.1317158E+02
EXIT E  CPUTIME= 6.7 I/O(DIR,BUF)= 0 0
** BEGIN EKS ** DATA SPACE= 200000 WORDS
E41 COMPLETED
EXIT EKS CPUTIME= 6.7 I/O(DIR,BUF)= 0 0
OKAY 1
** BEGIN AUS ** DATA SPACE= 200000 WORDS
MACR COMPLETED.
MACR COMPLETED.
MACR COMPLETED.
MACR COMPLETED.

OKAY 1.2
OKAY 1.4
njsect = 43
nsect[1] = 1
MACR COMPLETED.
EXIT AUS CPUTIME= 9.0 I/O(DIR,BUF)= 0 0
<CL> lv:0 t:615.8 SECTL[1] 10.000000
OKAY 2 2 2
OKAY 3 3 3
EXIT TAB CPUTIME= 42.4 I/O(DIR,BUF)= 0 0
** BEGIN ELD ** DATA SPACE= 200000 WORDS
EXIT ELD CPUTIME= 13.8 I/O(DIR,BUF)= 0 0
** BEGIN TOPO ** DATA SPACE= 200000 WORDS
<DM> OPEN, Ldi: 25, File: NS.L25 , Attr: scratch, Block I/O
<DM> OPEN, Ldi: 26, File: NS.L26 , Attr: scratch, Block I/O
NO. OF 4-NODE ELEMENTS= 44
OTOTAL NO. OF ELEMENTS= 44
OMAXCON, MAXSUB, ILMAX= 1872 1400 52
OKSIZE,NR5,LR5= 52 2 896
OMAXCON, MAXSUB, ILMAX= 1863 1400 52
OSIZE INDEX= 210, IC1, IC2= 4905 702, NR4= 4
<DM> CLOSE, Ldi: 25, File: NS.L25
<DM> CLOSE, Ldi: 26, File: NS.L26
EXIT TOPO CPUTIME= 14.9 I/O(DIR,BUF)= 0 0
** BEGIN E ** DATA SPACE= 200000 WORDS
T= 0.10000E-19-0.10000E-02 0.10000E-04 0.10000E-04
0.20000E+02 0.10000E-03 0.10000E-03 0.10000E-03
ERROR LEVELS= 2 2 0 2 2 2 2
0
0
L, VOL OR STRUCTURAL NON-STRUCTURAL
TYPE GROUP AREA SUM WEIGHT WEIGHT
E41 1 0.329000E+03 0.000000E+00 0.000000E+00

0
TOTAL                0.000000E+00 0.000000E+00

```

```

TOTAL 4-NODE@ 0.3290000E+03
EXIT E CPUTIME= 11.7 I/O(DIR,BUF)= 0 0
** BEGIN EKS ** DATA SPACE= 200000 WORDS
E41 COMPLETED
EXIT EKS CPUTIME= 15.9 I/O(DIR,BUF)= 0 0
OKAY 1
** BEGIN AUS ** DATA SPACE= 200000 WORDS
MACR COMPLETED.
MACR COMPLETED.
MACR COMPLETED.
MACR COMPLETED.

OKAY 1.2
OKAY 1.4
njsect = 43
nsect[2] = 1
MACR COMPLETED.
EXIT AUS CPUTIME= 12.1 I/O(DIR,BUF)= 0 0
<CL> lv:0 t:615.8 SECTL[1] 10.000000
OKAY 2 2 2
OKAY 3 3 3
** BEGIN AUS ** DATA SPACE= 200000 WORDS
MACR COMPLETED.
MACR COMPLETED.
MACR COMPLETED.
MACR COMPLETED.

EXIT AUS CPUTIME= 954.2 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
TABL COMPLETED.
TABL COMPLETED.
TABL COMPLETED.

EXIT AUS CPUTIME= 214.2 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
DEFI COMPLETED.
DEFI COMPLETED.
TABL COMPLETED.

EXIT AUS CPUTIME= 25.3 I/O(DIR,BUF)= 0 0
START 78 4,5,6

78 JOINTS.
OACTIVE JOINT MOTION COMPONENTS= 1 2 3
EXIT TAB CPUTIME= 10.7 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
MACR COMPLETED.
# of elements in substructure 1 - 16
NJ of DEF dataset is 80
MACR COMPLETED.
# of elements of substructure 2 - 44
** BEGIN ELD ** DATA SPACE= 200000 WORDS
EXIT ELD CPUTIME= 20.7 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
TABL COMPLETED.

```

```

EXIT AUS CPUTIME= 4.2 I/O(DIR,BUF)= 0 0
** BEGIN TOPO ** DATA SPACE= 200000 WORDS
<DM> OPEN, Ldi: 25, File: NS.L25 , Attr: scratch, Block I/O
<DM> OPEN, Ldi: 26, File: NS.L26 , Attr: scratch, Block I/O
NO. OF 4-NODE ELEMENTS= 60
OTOTAL NO. OF ELEMENTS= 60
OMAXCON, MAXSUB, ILMAX= 1872 1400 52
OKSIZE,NR5,LR5= 52 3 896
OMAXCON, MAXSUB, ILMAX= 1863 1400 52
OSIZE INDEX= 210, IC1, IC2= 5923 872, NR4= 4
<DM> CLOSE, Ldi: 25, File: NS.L25
<DM> CLOSE, Ldi: 26, File: NS.L26
EXIT TOPO CPUTIME= 21.2 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
DEFI COMPLETED.
DEFI COMPLETED.
UNIO COMPLETED.
EXIT AUS CPUTIME= 15.1 I/O(DIR,BUF)= 0 0
** BEGIN K ** DATA SPACE= 200000 WORDS
EXIT K CPUTIME= 28.0 I/O(DIR,BUF)= 0 0
** BEGIN INV ** DATA SPACE= 200000 WORDS
ONSING,NNEG= 0 0
EXIT INV CPUTIME= 20.7 I/O(DIR,BUF)= 0 0
** BEGIN AUS ** DATA SPACE= 200000 WORDS
SYSV COMPLETED.
EXIT AUS CPUTIME= 3.5 I/O(DIR,BUF)= 0 0
** BEGIN SSOL ** DATA SPACE= 200000 WORDS
<DM> OPEN, Ldi: 26, File: NS.L26 , Attr: scratch, Block I/O

```

```

0 CASE      F*U      U*KU      ERR

      1 0.3532554E+00 0.3532554E+00 -0.1687296E-06
<DM> CLOSE, Ldi: 26, File: NS.L26
EXIT SSOL CPUTIME= 18.1 I/O(DIR,BUF)= 0 0
** BEGIN DCU ** DATA SPACE= 200000 WORDS
0
EXIT DCU CPUTIME= 0.2 I/O(DIR,BUF)= 0 0
** BEGIN VPRT ** DATA SPACE= 200000 WORDS
OLIB READ ERROR, NU,L,KORE,IERR,IOP= 1 0 0 -1 10
NAME= CASE TITL 11263747405
1STATIC DISPLACEMENTS. ID= 1/ 1/ 1
0JOINT      1      2      3
1 0.123E-01 -0.173E-02 0.000E+00*
2 0.134E-01 -0.107E-02 0.000E+00*
3 0.152E-01 -0.716E-03 0.000E+00*
4 0.174E-01 -0.115E-02 0.000E+00*
5 0.184E-01 -0.173E-02 0.000E+00*
6 0.174E-01 -0.232E-02 0.000E+00*
7 0.152E-01 -0.275E-02 0.000E+00*
8 0.134E-01 -0.240E-02 0.000E+00*
9 0.123E-01 -0.173E-02 0.000E+00*
10 0.136E-01 -0.134E-02 0.000E+00*
11 0.154E-01 -0.478E-03 0.000E+00*

```

12	0.170E-01	-0.133E-02	0.000E+00*
13	0.186E-01	-0.173E-02	0.000E+00*
14	0.170E-01	-0.214E-02	0.000E+00*
15	0.154E-01	-0.299E-02	0.000E+00*
16	0.136E-01	-0.213E-02	0.000E+00*
17	0.122E-01	-0.173E-02	0.000E+00*
18	0.130E-01	-0.110E-02	0.000E+00*
19	0.154E-01	-0.584E-03	0.000E+00*
20	0.177E-01	-0.109E-02	0.000E+00*
21	0.185E-01	-0.173E-02	0.000E+00*
22	0.177E-01	-0.238E-02	0.000E+00*
23	0.154E-01	-0.288E-02	0.000E+00*
24	0.130E-01	-0.237E-02	0.000E+00*
25	0.307E-01	-0.348E-02	0.000E+00*
26	0.264E-01	-0.340E-02	0.000E+00*
27	0.219E-01	-0.340E-02	0.000E+00*
28	0.174E-01	-0.371E-02	0.000E+00*
29	0.154E-01	-0.379E-02	0.000E+00*
30	0.134E-01	-0.371E-02	0.000E+00*
31	0.882E-02	-0.340E-02	0.000E+00*
32	0.430E-02	-0.341E-02	0.000E+00*
33	0.000E+00*	-0.347E-02	0.000E+00*
34	0.307E-01	-0.311E-02	0.000E+00*
35	0.264E-01	-0.303E-02	0.000E+00*
36	0.220E-01	-0.300E-02	0.000E+00*
37	0.174E-01	-0.332E-02	0.000E+00*
38	0.154E-01	-0.343E-02	0.000E+00*
39	0.133E-01	-0.332E-02	0.000E+00*
40	0.873E-02	-0.300E-02	0.000E+00*
41	0.433E-02	-0.303E-02	0.000E+00*
42	0.000E+00*	-0.310E-02	0.000E+00*
43	0.307E-01	-0.233E-02	0.000E+00*

1

44	0.264E-01	-0.232E-02	0.000E+00*
45	0.222E-01	-0.226E-02	0.000E+00*
46	0.853E-02	-0.226E-02	0.000E+00*
47	0.435E-02	-0.232E-02	0.000E+00*
48	0.000E+00*	-0.233E-02	0.000E+00*
49	0.307E-01	-0.173E-02	0.000E+00*
50	0.264E-01	-0.173E-02	0.000E+00*
51	0.221E-01	-0.173E-02	0.000E+00*
52	0.867E-02	-0.173E-02	0.000E+00*
53	0.430E-02	-0.173E-02	0.000E+00*
54	0.000E+00*	-0.173E-02	0.000E+00*
55	0.307E-01	-0.113E-02	0.000E+00*
56	0.264E-01	-0.115E-02	0.000E+00*
57	0.222E-01	-0.121E-02	0.000E+00*
58	0.853E-02	-0.121E-02	0.000E+00*
59	0.435E-02	-0.115E-02	0.000E+00*
60	0.000E+00*	-0.114E-02	0.000E+00*
61	0.307E-01	-0.362E-03	0.000E+00*
62	0.264E-01	-0.438E-03	0.000E+00*
63	0.220E-01	-0.469E-03	0.000E+00*
64	0.174E-01	-0.151E-03	0.000E+00*

65	0.154E-01	-0.341E-04	0.000E+00*
66	0.133E-01	-0.153E-03	0.000E+00*
67	0.873E-02	-0.469E-03	0.000E+00*
68	0.433E-02	-0.433E-03	0.000E+00*
69	0.000E+00*	-0.372E-03	0.000E+00*
70	0.307E-01	0.125E-04	0.000E+00*
71	0.264E-01	-0.640E-04	0.000E+00*
72	0.219E-01	-0.701E-04	0.000E+00*
73	0.174E-01	0.243E-03	0.000E+00*
74	0.154E-01	0.325E-03	0.000E+00*
75	0.134E-01	0.242E-03	0.000E+00*
76	0.882E-02	-0.701E-04	0.000E+00*
77	0.430E-02	-0.596E-04	0.000E+00*
78	0.000E+00*	0.000E+00*	0.000E+00*

OLIB READ ERROR, NU,L,KORE,IERR,IOP= 1 0 0 -1 10
NAME= CASE TITL 11263747405
1STATIC REACTIONS, FORCE ERRORS. ID= 1/ 1/ 1

JOINT	1	2	3
1	0.100E-06	0.109E-06	0.000E+00*
2	0.583E-06	-0.224E-06	0.000E+00*
3	0.982E-06	0.220E-06	0.000E+00*
4	0.659E-06	-0.384E-06	0.000E+00*
5	-0.584E-06	0.113E-05	0.000E+00*
6	0.177E-06	0.668E-06	0.000E+00*
7	0.695E-06	-0.108E-06	0.000E+00*
8	0.352E-06	-0.740E-06	0.000E+00*
9	0.140E-05	-0.858E-07	0.000E+00*
10	-0.967E-06	-0.569E-06	0.000E+00*
11	-0.231E-05	-0.168E-06	0.000E+00*
12	-0.252E-05	-0.122E-05	0.000E+00*
13	-0.177E-05	0.186E-05	0.000E+00*
14	-0.456E-05	-0.245E-08	0.000E+00*
15	-0.491E-06	-0.823E-06	0.000E+00*
16	0.240E-05	0.284E-06	0.000E+00*
17	-0.113E-05	-0.833E-06	0.000E+00*
18	0.318E-05	-0.154E-06	0.000E+00*
19	0.867E-06	0.162E-06	0.000E+00*
20	0.411E-05	-0.230E-06	0.000E+00*
21	0.488E-05	-0.138E-05	0.000E+00*
22	0.213E-05	0.139E-05	0.000E+00*
23	0.370E-06	-0.750E-06	0.000E+00*
24	-0.923E-06	0.103E-05	0.000E+00*
25	0.954E-06	0.149E-06	0.000E+00*
26	0.197E-06	-0.454E-06	0.000E+00*
27	0.974E-06	0.388E-06	0.000E+00*
28	0.414E-06	0.415E-06	0.000E+00*
29	-0.474E-06	0.488E-06	0.000E+00*
30	0.525E-06	0.123E-06	0.000E+00*
31	-0.307E-06	0.436E-06	0.000E+00*
32	-0.588E-07	0.813E-07	0.000E+00*
33	-0.620E+00*	-0.365E-07	0.000E+00*
34	-0.238E-06	0.453E-06	0.000E+00*
35	0.128E-05	-0.149E-06	0.000E+00*
36	0.350E-05	-0.867E-06	0.000E+00*

```

37 0.240E-05 -0.118E-05 0.000E+00*
38 -0.346E-05 -0.531E-06 0.000E+00*
39 0.154E-05 -0.723E-07 0.000E+00*
40 -0.751E-06 0.330E-06 0.000E+00*
41 -0.959E-07 -0.841E-06 0.000E+00*
42 -0.187E+01* -0.146E-06 0.000E+00*
43 -0.238E-05 0.464E-06 0.000E+00*

```

1

```

44 0.275E-05 0.106E-05 0.000E+00*
45 -0.655E-05 -0.226E-06 0.000E+00*
46 0.746E-06 -0.305E-06 0.000E+00*
47 0.911E-06 0.415E-06 0.000E+00*
48 -0.225E+01* 0.299E-06 0.000E+00*
49 -0.238E-05 0.104E-05 0.000E+00*
50 -0.330E-05 -0.676E-06 0.000E+00*
51 -0.697E-06 -0.187E-05 0.000E+00*
52 -0.477E-06 0.201E-06 0.000E+00*
53 -0.100E-05 0.341E-06 0.000E+00*
54 -0.202E+01* 0.592E-07 0.000E+00*
55 0.334E-05 -0.133E-06 0.000E+00*
56 -0.287E-05 0.136E-05 0.000E+00*
57 -0.159E-05 0.958E-06 0.000E+00*
58 -0.104E-05 -0.367E-06 0.000E+00*
59 -0.202E-06 -0.314E-06 0.000E+00*
60 -0.225E+01* -0.458E-07 0.000E+00*
61 0.715E-06 -0.646E-06 0.000E+00*
62 0.199E-05 -0.162E-05 0.000E+00*
63 0.291E-05 0.101E-05 0.000E+00*
64 -0.752E-06 0.849E-06 0.000E+00*
65 -0.190E-07 0.553E-06 0.000E+00*
66 0.436E-06 -0.145E-06 0.000E+00*
67 0.716E-07 -0.438E-06 0.000E+00*
68 -0.166E-06 0.946E-07 0.000E+00*
69 -0.187E+01* -0.121E-06 0.000E+00*
70 -0.119E-06 0.276E-07 0.000E+00*
71 -0.135E-05 0.967E-06 0.000E+00*
72 -0.175E-05 0.204E-06 0.000E+00*
73 0.854E-06 -0.544E-06 0.000E+00*
74 -0.754E-06 -0.336E-07 0.000E+00*
75 -0.101E-05 -0.187E-06 0.000E+00*
76 -0.411E-06 -0.285E-06 0.000E+00*
77 -0.190E-06 -0.155E-06 0.000E+00*
78 -0.620E+00* 0.493E-05* 0.000E+00*

```

```

EXIT VPRT CPUTIME= 8.3 I/O(DIR,BUF)= 0 0
<DM> CLOSE, Ldi: 1, File: PAN1.L01
<CL> CSS exhausted
ENDRUN called by CLIP

```


APPENDIX C

PROGRAM FOR ELASTO-PLASTIC STABILITY PROBLEM

The program written in PASCAL is divided into two parts. Part I is called PROGRAM COLUMN. This part of the program is down-loaded onto one of the NPR number of processors and this processor is designated as "Master Processor". Part II of the program is named PROGRAM TANGENT. This program is down-loaded onto the rest of the processors of the NPR number of processors. These (NPR-1) number of processors are called "Assistant Processors". The assistant processors are utilized to compute elastic-plastic cross-sectional properties of various cross sections along the length of the member which are communicated to the master processor. The master processor assembles the global stiffness matrix upon receipt of the cross sectional properties from assistants. The lateral displacement vector is computed on the master processor. Thus the operation is sequential between the assembly of global stiffness values and evaluation of the lateral displacements of the member. A detailed flow chart of the algorithm used and the concurrent computational procedure is outlined in Reference 6.

The input to the program consists of the member length, material constants, end rotational stiffness characteristics, initial out-of-straightness midspan amplitudes, cross-sectional dimensions, and residual stresses. The output from the program consists of axial load level, corresponding converged lateral displacements of the member, nondimensionalized determinant of the global stiffness matrix at each load level, and the computational times of each processor.

**ORIGINAL PAGE IS
OF POOR QUALITY**

PROGRAM FOR ELASTO-PLASTIC STABILITY PROBLEM

```
(*#WIDELIST, NO ASSERTS, NO TRACEBACK*)
(* $GLOBALOPT *)
```

PROGRAM COLUMN:

```
CONST  MAXIDX=255;  (*MAX NBR I/O INDEX TAG *)
        MAXREC=255;  (*MAX REC LENGTH FOR NBR I/O *)
        MAXNODE=36;  (*MAX NODE NUMBERS *)
        MAXDA=31;    (*MAX DATA AREA NUMBER *)
        MAXINT=32767;(*MAX INTEGER *)
        SYSFLAG=1;   (*SYSTEM FLAG *)
```

```
TYPE   NODE=1..MAXNODE;
        IDX=1..MAXIDX;
        RECLEN=1..MAXREC;
        DANUM=1..MAXDA;
        FLAG=0..7;
        ADDR=INTEGER;
        POSINT=1..MAXINT;
```

```
(* PASLIB EXTERNAL PROCEDURES AND FUNCTIONS FOLLOW *)
```

```
(*-----*)
```

```
(*----- TIMER PROCEDURES -----*)
```

```
FUNCTION XTIME:LONGINT;EXTERNAL;
```

```
PROCEDURE TSTART(T:POSINT);EXTERNAL;
```

```
FUNCTION TREAD:LONGINT;EXTERNAL;
```

```
PROCEDURE TSTOP;EXTERNAL;
```

```
(*----- TEXT OUTPUT PROCEDURES -----*)
```

```
PROCEDURE MSG(String:PACKED ARRAY[1..?] OF CHAR);EXTERNAL;
```

```
PROCEDURE MSGLN(String:PACKED ARRAY[1..?] OF CHAR);EXTERNAL;
```

```
PROCEDURE NXTLN;EXTERNAL;
```

```
PROCEDURE MSGI(I:INTEGER);EXTERNAL;
```

```
PROCEDURE MSGR(X:REAL);EXTERNAL;
```

```
PROCEDURE MSGI(I:LONGINT);EXTERNAL;
```

```
PROCEDURE ENDLN(N:POSINT);EXTERNAL;
```

```
(*-----*)
```

```
(*----- INTERACTIVE INPUT -----*)
```

```
(*-----*)
```

```
PROCEDURE QUERY;EXTERNAL;
```

```
FUNCTION RDI:INTEGER;EXTERNAL;
```

```
FUNCTION RDR:REAL;EXTERNAL;
```

```
(*-----*)
```

```
(*----- DATA AREAS -----*)
```

```
FUNCTION DAPTR(DA:DANUM):ADDR;EXTERNAL;
```

```
(*-----*)
```

```
(*----- FLOATING OPERATIONS -----*)
```

```
FUNCTION ADD(X,Y:REAL):REAL;EXTERNAL;
```

```

FUNCTION SUB(X,Y:REAL):REAL;EXTERNAL;
FUNCTION MULT(X,Y:REAL):REAL;EXTERNAL;
FUNCTION DIVD(X,Y:REAL):REAL;EXTERNAL;
FUNCTION NEG(X:REAL):REAL;EXTERNAL;
FUNCTION ABS95(X:REAL):REAL;EXTERNAL;
FUNCTION CMP(X,Y:REAL):REAL;EXTERNAL;
FUNCTION SQRT95(X:REAL):REAL;EXTERNAL;
(*)
(*----- FLOATING POINT CONSTANTS -----*)
FUNCTION MAX95:REAL;EXTERNAL;
FUNCTION MIN95:REAL;EXTERNAL;
(*----- FLOATING POINT CONVERSIONS -----*)
FUNCTION IFIX(X:REAL):INTEGER;EXTERNAL;
FUNCTION FLOATI(I:INTEGER):REAL;EXTERNAL;
FUNCTION CV9512(X:REAL):REAL;EXTERNAL;
FUNCTION VDP(N:POSINT; VAR A:ARRAY[1..?] OF REAL;
            VAR B:ARRAY[1..?] OF REAL):REAL;EXTERNAL;
(*----- NBR COMMUNICATIONS -----*)
PROCEDURE SEND2(N:NODE; INDEX:IDX; LOC:ADDR; NWORDS:RECLN);EXTERNAL;
PROCEDURE SEND2ALL(INDEX:IDX; LOC:ADDR; NWORDS:RECLN);EXTERNAL;
PROCEDURE RECV2(N:NODE; INDEX:IDX; LOC:ADDR; NWORDS:RECLN);EXTERNAL;
(*----- FLAG ROUTINES -----*)
PROCEDURE FLGEN(F:FLAG);EXTERNAL;
PROCEDURE FLOSET(F:FLAG);EXTERNAL;
PROCEDURE FLGRES(F:FLAG);EXTERNAL;
PROCEDURE BAR(F:FLAG);EXTERNAL;
FUNCTION ANY(F:FLAG):BOOLEAN;EXTERNAL;
(*----- PROCESSOR IDENTIFICATION -----*)

PROCEDURE PSCL$$;

CONST
  EDIM=3;
TYPE
  DA7=ARRAY[1..3] OF INTEGER;
  DA8=ARRAY[1..4] OF REAL;
VAR
  PB:REAL; NENTRY:INTEGER; DATA:ARRAY [1..4] OF REAL;
  NEW, IO, NPR, NSEC, NPROB, N, MDIM : INTEGER;
  ELASTIC : BOOLEAN;
  ZERO, ONE, TWO : REAL;
  DET1, DET, PINC, TOL1, TOL2, TOL3 : REAL;
  DALL7 :@DA7; DALL8 :@DA8;
  CLK1, CLK3 : LONGINT;
PROCEDURE GLOBAL(N, MDIM, NPROB, NSEC: INTEGER);

```

```

LABEL 10,20;

TYPE
  VECTOR=ARRAY[1..N] OF REAL;
  MATRIX=ARRAY[1..N,1..N] OF REAL;
VAR
  K:MATRIX; DIS,DI,D,F:VECTOR;
  UV:ARRAY[1..MDIM] OF REAL;
  COEF:ARRAY[1..14] OF REAL; IDSEC,ITER,TANIDX:INTEGER;
                                COUNT,CNTR,NITER,ISEC:INTEGER;
  READY:ARRAY[1..NPR,1..NPROB] OF BOOLEAN;
                                ICNT:INTEGER;
  (*****)

PROCEDURE MATINV(VAR A:MATRIX; N:INTEGER; VAR DET:REAL);
(* MATRIX INVERSION PROCEDURE *)
(* METHOD - JOHNSON(1) METHOD *)
(* A IS N X N SYMMETRIC MATRIX*)

  VAR
    IPV:ARRAY[1..N,1..3] OF INTEGER;
    NSWAP,L,I,ICOL,JCOL,IROW,JROW :INTEGER;
    TEMP,T,PIV,SWAP,AMAX :REAL;

  BEGIN
    DET:=ONE; TEMP:=A[1,1];
    FOR J:=1 TO N DO IPV[J,3]:=0;
      I:=1;
      REPEAT (* UNTIL DET=0 OR I>N *)
        AMAX:=ZERO;
        FOR J:=1 TO N DO
          BEGIN
            IF IPV[J,3]<>1 THEN
              BEGIN
                FOR KK:=1 TO N DO
                  BEGIN
                    IF IPV[KK,3]<>1 THEN
                      IF CMP(AMAX,ABS95(A[J,KK]))<0 THEN
                        BEGIN
                          IROW:=J; (* SAVE ROW & COLUMN *)
                          ICOL:=KK;
                          AMAX:=ABS95(A[J,KK]);
                        END; (* IF AMAX *)
                      END; (* FOR KK *)
                    END; (* IF IPV[J,3] *)
                  END;
                IF IPV[J,3]<>1 THEN
                  DET:=DET/A[J,I];
                END;
              END;
            IF IPV[J,3]=1 THEN
              TEMP:=TEMP/A[J,I];
            END;
          END;
        UNTIL DET=0 OR I>N;
      END;
    END;
  END;

```

```

END;      (* FOR J *)

(* IF PIVOT ELEMENT IS EQ 0 - DET = 0 *)
IF CMP(AMAX,TOL2)<0 THEN DET:=ZERO

ELSE
BEGIN
  IPV[ICOL,3]:=IPV[ICOL,3]+1;
  IPV[I,1]:=IROW;
  IPV[I,2]:=ICOL;
  (* INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL *)
  IF IROW<>ICOL THEN
    FOR J:=1 TO N DO
      BEGIN
        SWAP:=A[IROW,J];
        A[IROW,J]:=A[ICOL,J];
        A[ICOL,J]:=SWAP;
      END;      (* FOR J 1 TO N *)

  (* DIVIDE PIVOT ROW BY PIVOT ELEMENT *)
  PIV:=A[ICOL,ICOL];
  DET:=MULT(DET,DIVD(PIV,TEMP));
  A[ICOL,ICOL]:=ONE;
  FOR KK:=1 TO N DO
    A[ICOL,KK]:=DIVD(A[ICOL,KK],PIV);
  (* REDUCE THE NON-PIVOT ROW BY SUBTRACTION *)
  FOR L1:=1 TO N DO
    IF L1<>ICOL THEN      (* AVOID DIAGONAL *)
      BEGIN
        T:=A[L1,ICOL];
        A[L1,ICOL]:=ZERO;
        FOR KK:=1 TO N DO
          A[L1,KK]:=SUB(A[L1,KK],MULT(A[ICOL,KK],T))
        END;      (* IF L1 <> ICOL *)
      I:=I+1;      (* GO ON TO NEXT *)
    END;      (* ELSE IF AMAX<TOL *)
  UNTIL (I>N) OR CMP(DET,ZERO)=0;

  (* INTERCHANGE THE COLUMNS AND MODIFY DET *)
  NSWAP:=0;      (* THIS IS SIGN FLAG *)
  IF CMP(DET,ZERO)<>0 THEN BEGIN      (* DO ONLY IF DET.NE.0.0 *)
    FOR J:=1 TO N DO
      BEGIN
        L:=N-J+1;
        IF IPV[L,1]<>IPV[L,2] THEN

```

```

        BEGIN
        JROW:=IPV[L,1];
        JCOL:=IPV[L,2];
        NSWAP:=NSWAP+1; (* COUNT SWAPS *)
        FOR KK:=1 TO N DO
            BEGIN
            SWAP:=A[KK,JROW];
            A[KK,JROW]:=A[KK,JCOL];
            A[KK,JCOL]:=SWAP;
            END; (* FOR KK *)
        END; (* IF IPV *)
    END; (* FOR J *)

    IF ODD(NSWAP) THEN SWAP:=NEG(ONE)
    ELSE SWAP:=ONE;
    END; (* IF DET<>0.0 *)
    DET:=MULT(DET,SWAP);
    END; (* MATINV PROCEDURE *)
    (*****
    PROCEDURE SOLVE( N:INTEGER; VAR X:MATRIX; VAR SOLNS:VECTOR;
                    SUMS:VECTOR; VAR DET:REAL);

    BEGIN (* MAIN PROGRAM *)
    MATINV(X,N,DET); (* CALL MATINV PROCEDURE *)
    IF CMP(DET,ZERO)=0 THEN
        MSGLN('MATRIX IS SINGULAR')
    ELSE
        BEGIN
        FOR J:=1 TO N DO
            BEGIN
            SOLNS[J]:=ZERO; (* INITIALIZE *)
            SOLNS[J]:=VDP(N,X[J],SUMS);
            END; (* FOR J *)
        END; (* ELSE i.e. DET .NE. 0.0 *)
    END;

    PROCEDURE ASSEMBLE(IDSEC:INTEGER);

    VAR IK:INTEGER;

    BEGIN
    (*$ NO OPTIMIZE *)
    IF IDSEC<=3 THEN
    BEGIN
        IF IDSEC=1 THEN

```

```

BEGIN
  K[1,1]:=ADD(COEF[1],COEF[5]);
  K[1,2]:=COEF[3];
  K[1,3]:=SUB(COEF[1],COEF[5]);
  K[1,4]:=COEF[3];
  K[1,N-1]:=NEG(COEF[6]);
  K[1,N-3]:=COEF[6];
  F[1]:=COEF[7];
  K[2,1]:=COEF[8];
  K[2,2]:=ADD(COEF[10],COEF[12]);
  K[2,3]:=COEF[8];
  K[2,4]:=SUB(COEF[10],COEF[12]);
  K[2,N]:=NEG(COEF[13]);
  K[2,N-2]:=COEF[13];
  F[2]:=COEF[14];
END
ELSE
  IF IDSEC=2 THEN
    BEGIN
      K[3,1]:=COEF[5];
      K[3,3]:=SUB(COEF[2],COEF[5]);
      K[3,4]:=COEF[4];
      K[3,5]:=COEF[1];
      K[3,6]:=COEF[6];
      K[3,N-1]:=NEG(COEF[6]);
      K[3,N-3]:=COEF[6];
      F[3]:=COEF[7];
      K[4,2]:=COEF[12];
      K[4,3]:=COEF[9];
      K[4,4]:=SUB(COEF[11],COEF[12]);
      K[4,5]:=COEF[8];
      K[4,6]:=COEF[10];
      K[4,N]:=NEG(COEF[13]);
      K[4,N-2]:=COEF[13];
      F[4]:=COEF[14];
    END
  ELSE
    BEGIN
      K[5,1]:=COEF[5];
      K[5,3]:=SUB(COEF[1],COEF[5]);
      K[5,4]:=COEF[3];
      K[5,5]:=COEF[2];
      K[5,6]:=COEF[4];
      K[5,7]:=COEF[1];
      K[5,8]:=COEF[3];
    END
  END

```

```

      K[5,N-1]:=NEG(COEF[6]);
      K[5,N-3]:=COEF[6];
      F[5]:=COEF[7];
      K[6,2]:=COEF[12];
      K[6,3]:=COEF[8];
      K[6,4]:=SUB(COEF[10],COEF[12]);
      K[6,5]:=COEF[9];
      K[6,6]:=COEF[11];
      K[6,7]:=COEF[8];
      K[6,8]:=COEF[10];
      K[6,N]:=NEG(COEF[13]);
      K[6,N-2]:=COEF[13];
      F[6]:=COEF[14];
    END;      (* IDSEC=3 *)
END      (* IDSEC<=3 THEN *)
ELSE
  IF IDSEC>=NSEC-2 THEN
    BEGIN
      IK:=(EDIM-1)*IDSEC;
      IF IDSEC=NSEC-2 THEN
        BEGIN
          K[IK-1,1]:=COEF[5];
          K[IK-1,3]:=NEG(COEF[5]);
          K[IK-1,N-7]:=COEF[1];
          K[IK-1,N-6]:=COEF[3];
          K[IK-1,N-5]:=COEF[2];
          K[IK-1,N-4]:=COEF[4];
          K[IK-1,N-3]:=ADD(COEF[1],COEF[6]);
          K[IK-1,N-2]:=COEF[3];
          K[IK-1,N-1]:=NEG(COEF[6]);
          F[IK-1]:=COEF[7];
          K[IK,N-7]:=COEF[8];
          K[IK,N-6]:=COEF[10];
          K[IK,N-5]:=COEF[9];
          K[IK,N-4]:=COEF[11];
          K[IK,N-3]:=COEF[8];
          K[IK,N-2]:=ADD(COEF[10],COEF[13]);
          K[IK,N]:=NEG(COEF[13]);
          K[IK,2]:=COEF[12];
          K[IK,4]:=NEG(COEF[12]);
          F[IK]:=COEF[14];
        END
      ELSE
        IF IDSEC=NSEC-1 THEN
          BEGIN

```



```

K[IK-1,N-5]:=COEF[1];
K[IK-1,N-4]:=COEF[3];
K[IK-1,N-3]:=ADD(COEF[2],COEF[6]);
K[IK-1,N-2]:=COEF[4];
K[IK-1,N-1]:=NEG(COEF[6]);
K[IK-1,1]:=COEF[5];
K[IK-1,3]:=NEG(COEF[5]);
F[IK-1]:=COEF[7];
K[IK,N-5]:=COEF[8];
K[IK,N-4]:=COEF[10];
K[IK,N-3]:=COEF[9];
K[IK,N-2]:=ADD(COEF[11],COEF[13]);
K[IK,N]:=NEG(COEF[13]);
K[IK,2]:=COEF[12];
K[IK,4]:=NEG(COEF[12]);
F[IK]:=COEF[14];
END
ELSE
  BEGIN
    K[IK-1,N-3]:=ADD(COEF[1],COEF[6]);
    K[IK-1,N-2]:=COEF[3];
    K[IK-1,N-1]:=SUB(COEF[1],COEF[6]);
    K[IK-1,N]:=COEF[3];
    K[IK-1,1]:=COEF[5];
    K[IK-1,3]:=NEG(COEF[5]);
    F[IK-1]:=COEF[7];
    K[IK,N-3]:=COEF[8];
    K[IK,N-2]:=ADD(COEF[10],COEF[13]);
    K[IK,N-1]:=COEF[8];
    K[IK,N]:=SUB(COEF[10],COEF[13]);
    K[IK,2]:=COEF[12];
    K[IK,4]:=NEG(COEF[12]);
    F[IK]:=COEF[14];
  END;
END (* IDSEC>=NSEC-2 THEN *)
ELSE
  BEGIN
    IK:=(EDIM-1)*IDSEC;
    K[IK-1,1]:=COEF[5];
    K[IK-1,3]:=NEG(COEF[5]);
    K[IK-1,N-3]:=COEF[6];
    K[IK-1,N-1]:=NEG(COEF[6]);
    K[IK-1,2*(IDSEC-1)-1]:=COEF[1];
    K[IK-1,2*(IDSEC-1)]:=COEF[3];
    K[IK-1,2*IDSEC-1]:=COEF[2];

```

```

K[IK-1,2*IDSEC]:=COEF[4];
K[IK-1,2*(IDSEC+1)-1]:=COEF[1];
K[IK-1,2*(IDSEC+1)]:=COEF[3];
F[IK-1]:=COEF[7];
K[IK,2]:=COEF[12];
K[IK,4]:=NEG(COEF[12]);
K[IK,N-2]:=COEF[13];
K[IK,N]:=NEG(COEF[13]);
K[IK,2*(IDSEC-1)-1]:=COEF[8];
K[IK,2*(IDSEC-1)]:=COEF[10];
K[IK,2*IDSEC-1]:=COEF[9];
K[IK,2*IDSEC]:=COEF[11];
K[IK,2*(IDSEC+1)-1]:=COEF[8];
K[IK,2*(IDSEC+1)]:=COEF[10];
F[IK]:=COEF[14];
END;      (* IDSEC other than (<=3) (>=NSEC-2)      *)
END;(*      Assemble end      *)

BEGIN
  NENTRY:=0; ICNT:=0; CNTR:=0; CLK3:=0;
  ELASTIC:=TRUE;
  FOR I:= 1 TO N DO
    BEGIN
      FOR J:= 1 TO N DO K[I,J]:=ZERO;
      D[I]:=ZERO;
      DI[I]:=ZERO;
      F[I]:=ZERO;
      K[I,I]:=ONE
    END;
    FOR I:=1 TO MDIM DO UV[I]:=ZERO;
  REPEAT      (* SMALL PINC *)
  REPEAT      (* UNTIL CAPACITY *)
    PB:=ADD(PB,PINC);
    NITER:=0;
    REPEAT (* UNTIL UV CONVERGES *)
      DATA[1]:=SUB(D[4],D[2]);
      DATA[2]:=SUB(D[N-2],D[N]);
      DATA[3]:=SUB(D[3],D[1]);
      DATA[4]:=SUB(D[N-3],D[N-1]); (* Terms to set Mbx,Mtx,..etc. *)
      CLK1:=TREAD;SEND2ALL(1,LOCATION(PB),11);CLK3:=CLK3+TREAD-CLK1;
      NEW:=1;
      ISEC:=0;
      FOR I:= 1 TO NPR DO
        BEGIN
          FOR J:= 1 TO MDIM DO UV[J]:=D[ISEC+J];

```

```

      ISEC:=ISEC+MDIM;CLK1:=TREAD;
      SEND2(I,2,LOCATION(UV),2*MDIM);CLK3:=CLK3+TREAD-CLK1
    END;
    CLK1:=TREAD;
    BAR(2);
    CLK3:=CLK3+TREAD-CLK1;
    FOR I:=1 TO N DO  FOR J:=1 TO N DO  K[I,J]:=ZERO;
    FOR I:=1 TO NPR DO  FOR J:=1 TO NPROB DO  READY[I,J]:=FALSE;
    COUNT:=0;
    WHILE (COUNT<NSEC) DO  (* Asynchronous Communication *)
      FOR I:= 1 TO NPR DO
        FOR J:= 1 TO NPROB DO
          IF NOT READY[I,J] THEN
            BEGIN
              CLK1:=TREAD;
              RECV2(I,J,LOCATION(COEF),31);CLK3:=CLK3+TREAD-CLK1;
              IF ITER>ICNT THEN
                BEGIN
                  IF TANIDX=1 THEN GOTO 20;
                  READY[I,J]:=TRUE;
                  COUNT:=COUNT+1;
                  IF IO<=5 THEN MSGI(IDSEC);
                  ASSEMBLE(IDSEC)
                END;
              END;
            END;
          IF IO<=3 AND ANY(5) THEN
            BEGIN
              NXTLN;
              FOR I:=1 TO N DO  BEGIN
                FOR J:=1 TO N DO  BEGIN MSGR(K[I,J]);MSG('  ')  END;
                                ENDLN(2);  END;
              END;
            END;
          IF NENTRY=0 THEN
            BEGIN
              SOLVE(N,K,D,F,DET);
              DETI:=DET;  DET:=DIVD(DET,DETI);
              NENTRY:=1;
              FOR I:=1 TO N DO  DIC[I]:=F[I];
            END
          ELSE
            BEGIN
              SOLVE(N,K,D,F,DET);
              DET:=DIVD(DET,DETI);
              IF IO<=4 THEN BEGIN NXTLN;MSGR(DET);NXTLN;  END;
              IF CMP(DET,TOL2)<=0 THEN GOTO 20;
            END
          END
        END
      END
    END
  END

```

```

        CNTR:=0;
        FOR I:=1 TO N DO
        BEGIN
            IF CMP(ABS95(SUB(F[I],DI[I])),TOL1)<=0 THEN CNTR:=CNTR+1;
        END;
    END;
    IF (CNTR<N) THEN FOR I:=1 TO N DO DI[I]:=F[I];
    IF NOT ANY(5) THEN NEW:=0 ELSE IF CNTR=N THEN NEW:=0 ELSE NEW:=1;
    CLK1:=TREAD;SEND2ALL(3,LOCATION(NEW),1);BAR(7);CLK3:=CLK3+TREAD-CLK1;
    ICNT:=ICNT+1;
    NITER:=NITER+1;
    UNTIL (CNTR=N) OR (NITER=8) ;
    IF NITER>=8 THEN BEGIN MSG('NO GLOBAL CONVERGENCE'); GOTO 20 END;
    IF IO<=5 THEN
    BEGIN
        NXTLN;MSG('AXIAL LOAD=');MSGR(PB);
        NXTLN;MSGLN('DISPLACEMENT VECTOR');
        FOR I:=1 TO N DO BEGIN MSGR(DI[I]);MSG(' ');END;NXTLN;
    END;
    FOR I:=1 TO N DO DIS[I]:=DI[I];
    IF IO>=7 THEN BEGIN MSG('AXIAL LOAD=');MSGR(PB);NXTLN; END;
    MSG('Determinant of Global Stiffness Matrix =');MSGR(DET);NXTLN;
    UNTIL CMP(DET,TOL2)<=0;
20: ;
    IF TANIDX=1 OR CMP(DET,TOL2)<=0 THEN ICNT:=ICNT+1;
    IF NITER<8 THEN
        IF NEW=1 THEN BEGIN CLK1:=TREAD;
                        SEND2ALL(3,LOCATION(NEW),1);BAR(7);
                        CLK3:=CLK3+TREAD-CLK1; END;
    FOR I:=1 TO N DO DI[I]:=DIS[I];
    PB:=SUB(PB,PINC); (* Retain nonincremented PB *)
    PINC:=DIVD(PINC,TWO);
    UNTIL CMP(PINC,TOL3)<=0;
10: FLGSET(6);
    BAR(2);
    NXTLN;
    END; (* PROCEDURE GLOBAL *)

    BEGIN (* MAIN PROGRAM *)
    FLGEN(2);FLGEN(5);FLGEN(6);FLGEN(7);
    FLGRES(2);FLGRES(5);FLGRES(6);FLGRES(7);BAR(SYSFLAG);
    ZERO:=CV9512(0.0); ONE:=CV9512(1.0); TWO:=CV9512(2.0);
    DALL7::ADDR:=DAPTR(7); DALL8::ADDR:=DAPTR(8);
    NSEC:=DALL7@[1]; NPR:=DALL7@[2]; IO:=DALL7@[3];
    TOL1:=CV9512(DALL8@[1]); TOL2:=CV9512(DALL8@[2]);

```

```

TOL3:=CV9512(DALL8@[3]);
PING:=CV9512(DALL8@[4]); PB:=ZERO;
TSTART(1);
MSG('CROSS SECTIONS=');MSGI(NSEC);
MSG('    PROCESSORS=');MSGI(NPR+1); NXTLN;
NPROB:=NSEC DIV NPR;
MDIM:=NPROB*(EDIM-1);
N:=NSEC*(EDIM-1);
GLOBAL(N,MDIM,NPROB,NSEC);
TSTOP;
ENDLN(2);MSG('PROCESSING TIME =');MSGL(TREAD);NXTLN;
NXTLN;MSG('EXECUTION TIME =');MSGL(TREAD-CLK3);NXTLN;
END;

BEGIN (*NO OBJECT*)
END.

```

```
(* $WIDELIST, NO ASSERTS, NO TRACEBACK *)
(* $GLOBALOPT *)
```

```
PROGRAM TANGENT;
```

```
CONST MAXIDX=255; (*MAX NBR I/O INDEX TAG *)
      MAXREC=255; (*MAX REC LENGTH FOR NBR I/O *)
      MAXNODE=36; (*MAX NODE NUMBERS *)
      MAXDA=31;   (*MAX DATA AREA NUMBER *)
      MAXINT=32767; (*MAX INTEGER *)
      SYSFLAG=1;  (*SYSTEM FLAG *)
```

```
TYPE  NODE=1..MAXNODE;
      IDX=1..MAXIDX;
      RELEN=1..MAXREC;
      DANUM=1..MAXDA;
      FLAG=0..7;
      ADDR=INTEGER;
      POSINT=1..MAXINT;
```

```
(* PASLIB EXTERNAL PROCEDURES AND FUNCTIONS FOLLOW *)
```

```
(*-----*)
```

```
(*----- TIMER PROCEDURES -----*)
```

```
FUNCTION XTIME:LONGINT;EXTERNAL;
```

```
PROCEDURE TSTART(T:POSINT);EXTERNAL;
```

```
FUNCTION TREAD:LONGINT;EXTERNAL;
```

```
PROCEDURE TSTOP;EXTERNAL;
```

```
(*----- TEXT OUTPUT PROCEDURES -----*)
```

```
PROCEDURE MSG(String:PACKED ARRAY[1..?] OF CHAR);EXTERNAL;
```

```
PROCEDURE MSGLN(String:PACKED ARRAY[1..?] OF CHAR);EXTERNAL;
```

```
PROCEDURE NXTLN;EXTERNAL;
```

```
PROCEDURE MSGI(I:INTEGER);EXTERNAL;
```

```
PROCEDURE MSGR(X:REAL);EXTERNAL;
```

```
PROCEDURE MSGI(I:LONGINT);EXTERNAL;
```

```
PROCEDURE ENDLN(N:POSINT);EXTERNAL;
```

```
(* *)
```

```
(*----- INTERACTIVE INPUT -----*)
```

```
(* *)
```

```
PROCEDURE QUERY;EXTERNAL;
```

```
FUNCTION RDI:INTEGER;EXTERNAL;
```

```
FUNCTION RDR:REAL;EXTERNAL;
```

```
(* *)
```

```
(*----- DATA AREAS -----*)
```

```
FUNCTION DAPTR(DA:DANUM):ADDR;EXTERNAL;
```

```
(* *)
```

```
(*----- FLOATING OPERATIONS -----*)
```

```
FUNCTION ADD(X,Y:REAL):REAL;EXTERNAL;
```

```

FUNCTION SUB(X,Y:REAL):REAL;EXTERNAL;
FUNCTION MULT(X,Y:REAL):REAL;EXTERNAL;
FUNCTION DIVD(X,Y:REAL):REAL;EXTERNAL;
FUNCTION NEG(X:REAL):REAL;EXTERNAL;
FUNCTION ABS95(X:REAL):REAL;EXTERNAL;
FUNCTION CMP(X,Y:REAL):REAL;EXTERNAL;
FUNCTION SQRT95(X:REAL):REAL;EXTERNAL;
(*)
(*----- FLOATING POINT CONSTANTS -----*)
FUNCTION MAX95:REAL;EXTERNAL;
FUNCTION MIN95:REAL;EXTERNAL;
(*----- FLOATING POINT CONVERSIONS -----*)
FUNCTION IFIX(X:REAL):INTEGER;EXTERNAL;
FUNCTION FLOATI(I:INTEGER):REAL;EXTERNAL;
FUNCTION CV9512(X:REAL):REAL;EXTERNAL;
FUNCTION VDP(N:POSINT; VAR A:ARRAY[1..?] OF REAL;
            VAR B:ARRAY[1..?] OF REAL):REAL;EXTERNAL;
FUNCTION SINE(X:REAL):REAL;EXTERNAL;
(*----- NBR COMMUNICATIONS -----*)
PROCEDURE SEND2(N:NODE; INDEX:IDX; LOC:ADDR; NWORDS:RECLN);EXTERNAL;
PROCEDURE SEND2ALL(INDEX:IDX; LOC:ADDR; NWORDS:RECLN);EXTERNAL;
PROCEDURE RECV(N:NODE; LOC:ADDR; NWORDS:RECLN);EXTERNAL;
PROCEDURE RECV2(N:NODE; INDEX:IDX; LOC:ADDR; NWORDS:RECLN);EXTERNAL;
(*-----FLAG ROUTINES-----*)
PROCEDURE FLGEN(F:FLAG);EXTERNAL;
PROCEDURE FLGSET(F:FLAG);EXTERNAL;
PROCEDURE FLGRES(F:FLAG);EXTERNAL;
FUNCTION ANY(F:FLAG):BOOLEAN;EXTERNAL;
PROCEDURE BAR(F:FLAG);EXTERNAL;
(*----- PROCESSOR IDENTIFICATION -----*)
FUNCTION PSELF:NODE;EXTERNAL;
FUNCTION LSELF:NODE;EXTERNAL;

PROCEDURE PSCL$$;

CONST
    EDIM=3;
TYPE
    TANDIM=1..EDIM;
    TANMAT=ARRAY[TANDIM,TANDIM] OF REAL;
    VECTOR2=ARRAY[TANDIM] OF REAL;
    DA4=ARRAY[1..9] OF INTEGER;
    DAS=ARRAY[1..16] OF REAL;

VAR

```

```

S:TANMAT;    EL:BOOLEAN;
PB:REAL; NENTRY:INTEGER; DATA:ARRAY[1..4] OF REAL; ZBAR:REAL;
                                NEW,IO,MDIM,NDIM,NSEC,NPR : INTEGER;
                                CLK8,CLK9 : LONGINT;
                                STRES,LOAD,LINC,LOADB,DEL : VECTOR2;
                                PP,PR,MXP,MYP,MXRE,MYRE : REAL;
                                ZERO,ONE,TWO,FOUR,TWLVE,SIXTN,PI : REAL;
AR,ARND,IX,IXND,IY,IYND,ZX,ZXND,ZY,ZYND,RXND,RYND : REAL;
EBW,EBT,EDW,EAB,EAD,EDT,XRTB,XRTD,YRCB,YRCD : REAL;
IDX1,IDX2,IPR,NPROB,NCNTR,NS,NB,ND,NTB,NTD,CNTR,CNT2 : INTEGER;
B,D,T,E,SIGY,SIGRC,SIGRT,SIGRTM,SIGR,ER : REAL;
                                DET,RC,RT : REAL;
                                UINT,VINT,LENGTH,KBX,KBY,KTX,KTY,SEGL : REAL;
(*   Spring constants           *) CBX,CBY,CTX,CTY,C1,C2 : REAL;
                                DALL4:@DA4; DALL5 : @DA5;
                                TOL1,TOL2,TOL3,TOL4 : REAL;
(* SIGRTM MODIFIED TO a/c FOR c/c CALCULATIONS *)
PROCEDURE PROPERTY;

VAR
    B_2T,B_T,D_2T,D_T,BSQ,BCU,TSQ,TCU,DSQ,DCU:REAL;
    FNB,FNTB,FND,FNTD:REAL;

BEGIN
    (*
    (*----- CALCULATION OF GEOMETRIC PROPERTIES -----*)

    (*----- CALCULATION OF AREA & AREA ND -----*)
    AR:=ADD(MULT(TWO,MULT(D,T)),MULT(TWO,MULT(T,SUB(B,MULT(T,TWO)))));
    ARND:=MULT(FOUR,DIVD(AR,MULT(B,D)));
    (*MSG('AREA OF CROSS-SECTION AR=');
    MSGR(AR);MSGLN(' IN**2');
    MSG('NON-DIMENSIONAL AREA ARND=');
    MSGR(ARND);ENDLN(2);*)
    B_2T:=SUB(B,MULT(TWO,T));
    B_T:=SUB(B,T);
    D_2T:=SUB(D,MULT(TWO,T));
    D_T:=SUB(D,T);
    TSQ:=MULT(T,T);
    TCU:=MULT(TSQ,T);
    BSQ:=MULT(B,B);
    BCU:=MULT(B,BSQ);
    DSQ:=MULT(D,D);
    DCU:=MULT(D,DSQ);

```



```

(*----- CALCULATION OF IX & IXND -----*)
IX:=DIVD(MULT(B_2T,MULT(T,MULT(D_T,D_T))),FOUR);
IX:=ADD(IX,MULT(B_2T,DIVD(TCU,TWLVE)));
IX:=ADD(IX,MULT(T,DIVD(DCU,TWLVE)));
IX:=MULT(IX,TWO);
(*MSG('MOMENT OF INERTIA IX= ');
MSGR(IX);MSGLN(' IN**4');*)
IXND:=MULT(B,DCU);
IXND:=MULT(SIXTN,DIVD(IX,IXND));
(*MSG('NON-DIMENSIONAL MOMENT OF INERTIA IXND= ');
MSGR(IXND);ENDLN(2);*)
RXND:=SQRT95(DIVD(IXND,ARND));

(*----- CALCULATION OF IY & IYND -----*)
IY:=MULT(D_2T,MULT(T,MULT(B_T,DIVD(B_T,FOUR))));
IY:=ADD(IY,MULT(D_2T,DIVD(TCU,TWLVE)));
IY:=ADD(IY,MULT(BCU,DIVD(T,TWLVE)));
IY:=MULT(TWO,IY);
IYND:=MULT(SIXTN,DIVD(IY,MULT(BCU,D)));
(*MSG('MOMENT OF INERTIA IY= ');
MSGR(IY);MSGLN(' IN**4');
MSG('NON-DIMENSIONAL MOMENT OF INERTIA IYND= ');
MSGR(IYND);ENDLN(2);*)
RYND:=SQRT95(DIVD(IYND,ARND));

(*----- PLASTIC MODULI ZX,ZXND,ZY,ZYND -----*)
ZX:=ADD(MULT(B,MULT(T,DIVD(D_T,TWO))),MULT(T,MULT(D_2T,
DIVD(D_2T,FOUR))));
ZX:=MULT(TWO,ZX);
(*MSG('PLASTIC MODULUS ZX= ');
MSGR(ZX);MSGLN(' IN**3');*)
ZXND:=MULT(FOUR,MULT(TWO,DIVD(ZX,MULT(B,DSQ))));
(*MSG('NON-DIMENSIONAL PLASTIC MODULUS ZXND= ');
MSGR(ZXND);ENDLN(2);*)
ZY:=ADD(MULT(D,MULT(T,DIVD(B_T,TWO))),
MULT(T,MULT(B_2T,DIVD(B_2T,FOUR))));
ZY:=MULT(TWO,ZY);
(*MSG('PLASTIC MODULUS ZY= ');
MSGR(ZY);MSGLN(' IN**3');*)
ZYND:=MULT(FOUR,MULT(TWO,DIVD(ZY,MULT(D,BSQ))));
(*MSG('NON-DIMENSIONAL PLASTIC MODULUS ZYND= ');
MSGR(ZYND);ENDLN(2);*)

(*----- ELEMENTAL PROPERTIES -----*)

```

```

(*----- ALL QUANTITIES ARE NON-DIMENSIONAL -----*)
FNB:=FLOATI(NB);
FNTB:=FLOATI(NTB);
FND:=FLOATI(ND);
FNTD:=FLOATI(NTD);
EBW:=DIVD(TWO,FNB);
EBT:=MULT(TWO,DIVD(T,MULT(D,FNTB)));
EDW:=MULT(TWO,DIVD(T,MULT(B,FNTD)));
EDT:=DIVD(DL2T,FND);EDT:=DIVD(MULT(TWO,EDT),D);
(*MSGLN('MAJOR AXIS - HORIZONTAL');
MSGLN('ELEMENTAL PROPERTIES - NONDIMENSIONAL');
MSGLN('-----');
MSG('WIDTH OF ELEMENT IN FLANGES   EBW= ');
MSGR(EBW);NXTLN;
MSG('DEPTH OF ELEMENT IN FLANGES   EBT= ');
MSGR(EBT);NXTLN;
MSG('WIDTH OF ELEMENT IN WEB       EDW= ');
MSGR(EDW);NXTLN;
MSG('DEPTH OF ELEMENT IN WEB       EDT= ');
MSGR(EDT);ENDLN(3);*)
IF CMP(RT,ZERO)=0
THEN
  BEGIN
    XRTB:=ZERO;
    XRTD:=ZERO;
    YRCB:=ZERO;
    YRCD:=ZERO;
  END
ELSE
  BEGIN
    XRTB:=ADD(B,DL2T);
    XRTB:=MULT(MULT(RC,RT),XRTB);
    XRTB:=DIVD(XRTB,MULT(MULT(TWO,ADD(RC,RT)),ADD(RC,RT)));
    XRTD:=MULT(TWO,DIVD(XRTB,D));
    XRTB:=MULT(TWO,DIVD(XRTB,B));
    YRCB:=MULT(DIVD(RC,RT),XRTB);
    YRCD:=MULT(DIVD(RC,RT),XRTD);
    SIGRTM:=DIVD(MULT(RT,T),MULT(D,XRTD));
    SIGRTM:=ADD(RT,SIGRTM);
    (* MSG('SIGRTM= ');MSGR(SIGRTM);NXTLN;
    MSG('SIGRC= ');MSGR(SIGRC);ENDLN(3); *)
  END;
(*MSGLN('RESIDUAL STRESS DISTRIBUTION DIMENSIONS');
MSG('IN FLANGES   XRTB= ');MSGR(XRTB);MSG('   YRCB= ');MSGR(YRCB);NXTLN;
MSG('IN WEB       XRTD= ');MSGR(XRTD);MSG('   YRCD= ');MSGR(YRCD);NXTLN;*)

```

```

(* CALCULATE CONSTANTS *)
SEGL:=FLOATI(NSEC-1);
SEGL:=DIVD(LENGTH,SEGL);
C1:=MULT(DSQ,DIVD(E,SIGY));
C1:=DIVD(C1,MULT(MULT(SEGL,SEGL),FOUR)); (* Relates phi-ubar,vbar *)
C2:=MULT(BSQ,DIVD(E,SIGY));
C2:=DIVD(C2,MULT(MULT(SEGL,SEGL),FOUR));
(* CBX,CBY,CTX,CTY - Constants of Restrints *)
CBX:=MULT(MULT(TWO,FOUR),MULT(IX,SIGY));
CBX:=MULT(CBX,SEGL);      CBY:=CBX;
CBX:=DIVD(DSQ,CBX);
CBY:=DIVD(BSQ,CBY);
CTX:=MULT(CBX,KTX);
CBX:=MULT(CBX,KBX);
CTY:=MULT(CBY,KTY);
CBY:=MULT(CBY,KBY);
NXTLN;MSG('CBX,CTX,CBY,CTY,C1,C2  are ');
MSGR(CBX);MSGR(CTX);MSGR(CBY);MSGR(CTY);MSGR(C1);MSGR(C2);NXTLN;
END;      (*---      PROCEDURE PROPERTIES ---*)

```

```

(*)
PROCEDURE ROUTINE(N,M: INTEGER);
LABEL 10;
VAR
    COEF:ARRAY[1..14] OF REAL; IDSEC,ITER,TANIDX: INTEGER;
    STORED,STOREL:ARRAY[1..N] OF REAL;
    VCTRD,VCTRL:ARRAY[1..N] OF REAL;
    UV:ARRAY[1..M] OF REAL;
    UO,VO:REAL;
    ITR:ARRAY[1..NPR,1..NPROB] OF INTEGER;
(*----- PROCEDURE TO CALCULATE RESIDUAL STRESSES -----*)
PROCEDURE RESY(VAR SIGR,EPSR:REAL; Y:REAL);
VAR
    YD,FNB,FNTB,XRD:REAL;
BEGIN
    XRD:=ADD(XRTD,DIVD(T,D)); (* a/c FOR CORNER DEDUCTION *)
    YD:=ABS95(Y);
    IF CMP(YD,SUB(ONE,ADD(XRD,YRCD)))<=0 THEN
        BEGIN
            SIGR:=SIGRC;
            EPSR:=SIGR;
        END      (* IF - THEN *)
    ELSE

```

```

    BEGIN
      FNB:=SUB(ONE,YD);
      FNTB:=ABS95(SIGRC);
      IF CMP(RT,ZERO) <> 0
      THEN
        BEGIN
          SIGR:=DIVD(ADD(FNTB,SIGRTM),ADD(XRD,YRCD));
          SIGR:=SUB(SIGRTM,MULT(SIGR,FNB));
        END
      ELSE
        SIGR:=ZERO;
      EPSR:=SIGR;
    END;  (* IF - ELSE *)
  (* MSG('RESY');MSGR(SIGR);MSGR(EPSR);NXTLN; *)

END;  (* --- PROCEDURE END --- *)

PROCEDURE RESX(VAR SIGR,EPSR:REAL; X:REAL);

VAR
  XRB,XB,TT,T1:REAL;
BEGIN
  XRB:=ADD(XRTB,DIVD(T,B));  (* a/c FOR CORNER *)
  XB:=ABS95(X);
  IF CMP(XB,SUB(ONE,ADD(XRB,YRCB)))<=0 THEN
    BEGIN
      SIGR:=SIGRC;
      EPSR:=SIGR;
    END  (* IF - THEN *)
  ELSE
    BEGIN
      TT:=SUB(ONE,XB);
      T1:=ABS95(SIGRC);
      IF CMP(RT,ZERO) <> 0
      THEN
        BEGIN
          SIGR:=DIVD(ADD(T1,SIGRTM),ADD(XRB,YRCB));
          SIGR:=SUB(SIGRTM,MULT(TT,SIGR));
        END
      ELSE
        SIGR:=ZERO;
      EPSR:=SIGR;
    END;  (* IF - ELSE *)
  (* MSG('RESX');MSGR(SIGR);MSGR(EPSR);NXTLN; *)
END;  (* ----- PROCEDURE END ----- *)
PROCEDURE CHECK(VAR EL:BOOLEAN);FORWARD;
PROCEDURE ESTIFF(VAR S:TANMAT; VAR EL:BOOLEAN; DEL:VECTOR2;

```

```

        ARND,IXND,IYND:REAL; VAR PP,PR,MXP,MYP,MXRE,MYRE:REAL);FORWARD;
PROCEDURE SOLVE(X:TANMAT; SUMS:VECTOR2; VAR SOLNS:VECTOR2;
                VAR DET:REAL);FORWARD;
PROCEDURE TANGNT(LD,LOADB:VECTOR2; VAR DEL:VECTOR2; VAR S:TANMAT;
                VAR CNT2:INTEGER; U0,V0:REAL);
LABEL 10;
VAR
        LD1,SOLNS,DEL1:VECTOR2;
        CNT:INTEGER;
        CLK1,CLK2:LONGINT;
BEGIN
    CNT2:=0;
    CHECK(EL);
    IF (EL) THEN
        BEGIN
            SOLVE(S,LOADB,SOLNS,DET);
            FOR I:=1 TO EDIM DO DEL[I]:=SOLNS[I];
        END (* -- IF - THEN -- *)
    ELSE
        BEGIN
            FOR I:=1 TO EDIM DO DEL1[I]:=DEL[I];
            REPEAT (* -- UNTIL LD[1,2,&3] < TOL -- *)
                CNT:=0;
                CNT2:=CNT2+1;
                FOR I:=1 TO EDIM DO LD[I]:=SUB(LOADB[I],LD[I]);
                FOR I:=1 TO EDIM DO
                    IF CMP(ABS95(LD[I]),TOL1)<=0 THEN CNT:=CNT+1;
                IF CNT<EDIM THEN
                    BEGIN
                        SOLVE(S,LD,SOLNS,DET);
                        IF CMP(DET,TOL2)<=0 THEN GOTO 10;
                        FOR I:=1 TO EDIM DO DEL1[I]:=ADD(DEL1[I],SOLNS[I]);
                        (*---- CALCULATE STRESSES F1 --- *)
                        ESTIFF(S,EL,DEL1,ARND,IXND,IYND,PP,PR,MXP,MYP,MXRE,MYRE);
                        FOR I:=1 TO EDIM DO LD1[I]:=VDP(EDIM,S[I],DEL1);
                        LD1[1]:=ADD(LD1[1],ADD(PP,PR));
                        LD1[2]:=ADD(LD1[2],ADD(MXP,MXRE));
                        LD1[3]:=SUB(LD1[3],ADD(MYP,MYRE));
                        FOR I:=1 TO EDIM DO LD[I]:=LD1[I];
                    END; (* ----- ELSE CNVRG<>TRUE ----- *)
                IF CNT=EDIM THEN
                    FOR I:=1 TO EDIM DO DEL[I]:=DEL1[I];
                    IF CNT2=15 THEN MSGLN('NO CONVERGENCE IN 15 ITERATIONS')
                UNTIL (CNT=EDIM) OR (CNT2>=15)
            END; (* --- ELSE --- *)

```

```

(* MSGLN(' CONVERGED VALUES OF LOAD AND DISPLACEMENT ');
MSGLN('-----');
MSG(' ');MSG('LOAD VECTOR');
MSG(' ');MSGLN('DISPLACEMENT');
FOR I:=1 TO EDIM DO
  BEGIN
    NXTLN;MSGI(I);MSG(' ');MSGR(LOADB[I]);MSG(' ');MSGR(DEL[I]);
  END;
  *)
10: END; (* --- PROCEDURE TANGNT ---- *)

(* ----- PROCEDURE ESTIFF ----- *)
PROCEDURE ESTIFF; (* --(VAR S:TANMAT; VAR EL:BOOLEAN;
  ARND,IXND,IYND:REAL; VAR PP,PR,MXP,MYP,MXRE,MYRE:REAL)- *)
VAR
  CNT1,CNT2,MB,MD: INTEGER;
  SUM,ASUM,BSUM,CSUM,DSUM,ESUM, TX, TY, X, Y, TMP: REAL;
  DEL2,DEL3,B: REAL;
PROCEDURE STIFF;
(**$ NO OPTIMIZE *)
  BEGIN
    S[1,1]:=DIVD(SUM,ARND);
    S[1,2]:=DIVD(ASUM,ARND);
    PP:=DIVD(PP,ARND);
    PR:=DIVD(PR,ARND);
    MXP:=DIVD(MXP,IXND);
    MYP:=DIVD(MYP,IYND);
    MXRE:=DIVD(MXRE,IXND);
    MYRE:=DIVD(MYRE,IYND);
    S[1,3]:=NEG(DIVD(BSUM,ARND));
    S[3,1]:=NEG(DIVD(BSUM,IYND));
    S[3,2]:=NEG(DIVD(CSUM,IYND));
    S[3,3]:=DIVD(DSUM,IYND);
    S[2,3]:=NEG(DIVD(CSUM,IXND));
    S[2,2]:=DIVD(ESUM,IXND);
    S[2,1]:=DIVD(ASUM,IXND);
  END;
PROCEDURE STIFCAL(MB,NTB,IDX: INTEGER;
  EBT,EBW,EAB,DEL2,DEL3: REAL);
  LABEL 20,30;
  BEGIN
    (* IDX=0 USES RESX   IDX=1 USES RESY   *)
    CNT1:=1;
    30: FOR I:=1 TO NTB DO
      BEGIN

```

```

Y:=FLOATI(2*I-1);
Y:=SUB(ONE,MULT(Y,DIVD(EBT,TWO)));
IF CNT1>=2 THEN Y:=NEG(Y);
TY:=MULT(Y,DEL2);
CNT2:=1;
20:  FOR J:=1 TO MB DO
      BEGIN
        X:=FLOATI(2*J-1);
        X:=MULT(X,DIVD(EBW,TWO));
        IF CNT2>=2 THEN X:=NEG(X);
        IF IDX=0 THEN RESX(SIGR,ER,X)
          ELSE RESY(SIGR,ER,X);
        TX:=MULT(DEL3,X);
        IF IDX=0 THEN TX:=ADD(SUB(DEL[1],TX),ADD(TY,ER))
          ELSE TX:=ADD(SUB(DEL[1],TY),ADD(TX,ER));
        IF CMP(ABS95(TX),ONE)>=0 THEN
          BEGIN
            IF CMP(TX,ZERO)<0
              THEN
                BEGIN
                  PP:=ADD(PP,NEG(EAB));
                  IF IDX=0 THEN
                    BEGIN
                      MXP:=ADD(MXP,MULT(Y,NEG(EAB)));
                      MYP:=ADD(MYP,MULT(X,NEG(EAB)));
                    END
                  ELSE BEGIN
                      MXP:=ADD(MXP,MULT(X,NEG(EAB)));
                      MYP:=ADD(MYP,MULT(Y,NEG(EAB)));
                    END;
                  END
              ELSE
                BEGIN
                  PP:=ADD(PP,EAB);
                  IF IDX=0 THEN
                    BEGIN
                      MXP:=ADD(MXP,MULT(Y,EAB));
                      MYP:=ADD(MYP,MULT(X,EAB));
                    END
                  ELSE BEGIN
                      MXP:=ADD(MXP,MULT(X,EAB));
                      MYP:=ADD(MYP,MULT(Y,EAB));
                    END;
                  END;
                (* -- IF ELSE -- *)
              END
            (* IF TX>1.0 *)
          END
        END
      END

```

```

ELSE
  BEGIN
    SUM:=ADD(SUM,EAB);
    PR:=ADD(PR,MULT(SIGR,EAB));
    IF IDX=0 THEN
      BEGIN
        ASUM:=ADD(ASUM,MULT(Y,EAB));
        BSUM:=ADD(BSUM,MULT(X,EAB));
        CSUM:=ADD(CSUM,MULT(MULT(X,Y),EAB));
        DSUM:=ADD(DSUM,MULT(MULT(X,X),EAB));
        ESUM:=ADD(ESUM,MULT(MULT(Y,Y),EAB));
        MXRE:=ADD(MXRE,MULT(SIGR,MULT(Y,EAB)));
        MYRE:=ADD(MYRE,MULT(SIGR,MULT(X,EAB)));
      END
    ELSE BEGIN
      ASUM:=ADD(ASUM,MULT(X,EAB));
      BSUM:=ADD(BSUM,MULT(Y,EAB));
      CSUM:=ADD(CSUM,MULT(MULT(X,Y),EAB));
      DSUM:=ADD(DSUM,MULT(MULT(Y,Y),EAB));
      ESUM:=ADD(ESUM,MULT(MULT(X,X),EAB));
      MXRE:=ADD(MXRE,MULT(SIGR,MULT(X,EAB)));
      MYRE:=ADD(MYRE,MULT(SIGR,MULT(Y,EAB)));
    END;
  END; (* -- ELSE -- *)
END; (* --- FOR J --- *)
CNT2:=CNT2+1;
IF CNT2<=2 THEN GOTO 20;
END; (* --- FOR I --- *)
CNT1:=CNT1+1;
IF CNT1<=2 THEN GOTO 30;
END;
BEGIN
  FOR I:=1 TO EDIM DO
    BEGIN
      FOR J:=1 TO EDIM DO S[I,J]:=ZERO
    END;
    CHECK(EL); (* --- CHECK YIELDING OF CROSS-SECTION --- *)
    IF NOT (EL) THEN
      BEGIN
        MB:=NB DIV 2;
        MD:=ND DIV 2;
        PR:=ZERO;
        PP:=PR;
        MXP:=PP;
        MYP:=PP;

```



```

        MXRE:=PP;
        MYRE:=PP;

(*   note: P -VE, MX +VE, and MY -VE
        phiY BE POS. VALUE & phiX BE POS. VALUE   *)
        EAB:=MULT(EBW,EBT);
        EAD:=MULT(EDW,EDT);
        SUM:=ZERO;
        ASUM:=SUM;
        BSUM:=SUM;
        CSUM:=SUM;
        DSUM:=SUM;
        ESUM:=SUM;
        STIFCAL(MB,NTB,0,EBT,EBW,EAB,DEL[2],DEL[3]);
        STIFCAL(MD,NTD,1,EDW,EDT,EAD,DEL[3],DEL[2]);
        STIFF;
    END
    ELSE (* IF ELASTIC *)
    BEGIN
        FOR I:=1 TO EDIM DO
        BEGIN
            FOR J:=1 TO EDIM DO
            BEGIN
                IF I<>J THEN S[I,J]:=ZERO
                ELSE S[I,J]:=ONE;
            END;
        END;
        PR:=ZERO;
        PP:=PR;
        MXP:=PP;
        MYP:=PP;
        MXRE:=PP;
        MYRE:=PP;
    END; (* -- ELSE -- *)
END; (* -- PROCEDURE ESTIFF -- *)

(* ----- PROCEDURE TO CHECK YIELDING OF CROSS-SECTION ----- *)
(*                                                                 *)

PROCEDURE CHECK; (* --(VAR EL:BOOLEAN)-- *)

VAR
    X,Y,XRB,XRD:REAL;
    I:INTEGER;

```

```

BEGIN
  XRB:=ADD(XRTB,DIVD(T,B));
  XRD:=ADD(XRTD,DIVD(T,D));
  X:=ADD(XRB,YRCB);
  X:=SUB(ONE,X);
  X:=DIVD(X,EBW);
  I:=IFIX(X);
  I:=2*(I-1)+1;
  X:=FLOATI(I);
  X:=MULT(X,DIVD(EBW,TWO));
  IF CMP(DEL[3],ZERO)<0 THEN X:=NEG(X);
  RESX(SIGR,ER,X);
  X:=MULT(X,DEL[3]);
  Y:=SUB(ONE,DIVD(EBT,TWO));
  IF CMP(DEL[2],ZERO)>0 THEN Y:=NEG(Y);
  Y:=MULT(Y,DEL[2]);
  X:=ADD(SUB(DEL[1],X),ADD(Y,ER));
  IF CMP(ABS95(X),ONE) >=0
    THEN EL:=FALSE
    ELSE
      BEGIN
        Y:=ADD(XRD,YRCD);
        Y:=SUB(ONE,Y);
        Y:=DIVD(Y,EDT);
        I:=IFIX(Y);
        I:=2*(I-1)+1;
        Y:=FLOATI(I);
        Y:=MULT(Y,DIVD(EDT,TWO));
        IF CMP(DEL[2],ZERO)>0 THEN Y:=NEG(Y);
        RESY(SIGR,ER,Y);
        Y:=MULT(Y,DEL[2]);
        X:=SUB(ONE,DIVD(EDW,TWO));
        IF CMP(DEL[3],ZERO)<0 THEN X:=NEG(X);
        X:=MULT(X,DEL[3]);
        Y:=ADD(SUB(DEL[1],X),ADD(Y,ER));
        IF CMP(ABS95(Y),ONE) >=0
          THEN EL:=FALSE
          ELSE
            BEGIN
              X:=SUB(ONE,DIVD(EBW,TWO));
              IF CMP(DEL[3],ZERO)>0 THEN X:=NEG(X);
              Y:=SUB(ONE,DIVD(EDT,TWO));
              IF CMP(DEL[2],ZERO)<0 THEN Y:=NEG(Y);
              X:=MULT(X,DEL[3]);
              Y:=MULT(Y,DEL[2]);
            END
          END
        END
      END

```

C-2


```

IF CMP(AMAX,TOL2)<=0 THEN DET:=ZERO
ELSE
  BEGIN
    IPV[ICOL,3]:=IPV[ICOL,3]+1;
    IPV[I,1]:=IROW;
    IPV[I,2]:=ICOL;
    (* INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL *)
    IF IROW<>ICOL THEN
      FOR J:=1 TO N DO
        BEGIN
          SWAP:=A[IROW,J];
          A[IROW,J]:=A[ICOL,J];
          A[ICOL,J]:=SWAP;
        END;      (* FOR J 1 TO N *)
    (* DIVIDE PIVOT ROW BY PIVOT ELEMENT *)
    PIV:=A[ICOL,ICOL];
    DET:=MULT(DET,PIV);
    A[ICOL,ICOL]:=ONE;
    FOR K:=1 TO N DO
      A[ICOL,K]:=DIVD(A[ICOL,K],PIV);
    (* REDUCE THE NON-PIVOT ROW BY SUBTRACTION *)
    FOR L1:=1 TO N DO
      IF L1<>ICOL THEN      (* AVOID DIAGONAL *)
        BEGIN
          T:=A[L1,ICOL];
          A[L1,ICOL]:=ZERO;
          FOR K:=1 TO N DO
            A[L1,K]:=SUB(A[L1,K],MULT(A[ICOL,K],T))
          END;      (* IF L1 <> ICOL *)
        I:=I+1;      (* GO ON TO NEXT *)
      END;      (* ELSE IF AMAX<TOL *)
    UNTIL (I>N) OR CMP(DET,ZERO)=0;
    (* INTERCHANGE THE COLUMNS AND MODIFY DET *)
    NSWAP:=0;      (* THIS IS SIGN FLAG *)
    IF CMP(DET,ZERO)<>0 THEN BEGIN      (* DO ONLY IF DET.NE.0.0 *)
      FOR J:=1 TO N DO
        BEGIN
          L:=N-J+1;
          IF IPV[L,1]<>IPV[L,2] THEN
            BEGIN
              JROW:=IPV[L,1];
              JCOL:=IPV[L,2];
              NSWAP:=NSWAP+1; (* COUNT SWAPS *)
              FOR K:=1 TO N DO
                BEGIN

```

```

        SWAP:=A[K,JROW];
        A[K,JROW]:=A[K,JCOL];
        A[K,JCOL]:=SWAP;
    END;    (* FOR K *)
END;    (* IF IPV *)
END;    (* FOR J *)

IF ODD(NSWAP) THEN SWAP:=NEG(ONE)
    ELSE SWAP:=ONE;
END;    (* IF DET<>0.0 *)
DET:=MULT(DET,SWAP);
END;    (* MATINV PROCEDURE *)

PROCEDURE SOLVE;

VAR  SIZE:INTEGER;

BEGIN    (* MAIN PROGRAM *)
SIZE:=EDIM;
MATINV(X,SIZE); (* CALL MATINV PROCEDURE *)
IF CMP(DET,TOL2)<=0
    THEN MSGLN('MATRIX IS SINGULAR')
    ELSE
        BEGIN
            FOR J:=1 TO SIZE DO
                BEGIN
                    SOLNS[J]:=ZERO;    (* INITIALIZE *)
                    SOLNS[J]:=VDP(SIZE,X[J],SUMS);
                END;    (* FOR J *)
            END;    (* ELSE i.e. DET .NE. 0.0 *)
        END;
    (*=====*)
    BEGIN    (* Routine's main body starts here *)
        ITER:=0;FOR I:=1 TO NPR DO FOR J:=1 TO NPROB DO ITR[I,J]:=0;
        UINT:=DIVD(MULT(UINT,LENGTH),DIVD(B,TWO));
        VINT:=DIVD(MULT(VINT,LENGTH),DIVD(D,TWO));
        REPEAT    (* Until failure of column *)
            IPR:=0;    IDX1:=0; IDX2:=0;
            TANIDX:=0;
            CLK8:=TREAD;
            BAR(2);
            CLK9:=CLK9+TREAD-CLK8;
            IF ANY(6) THEN GOTO 10;
            CLK8:=TREAD;
            RECV2(NPR+1,1,LOCATION(PB),11);(*Recieve PB to DATA[4];DATA=M coef*)

```

```

RECV2(NPR+1,2,LOCATION(UV),2*MDIM); (*Recieve UBARs & VBARs *)
CLK9:=CLK9+TREAD-CLK8;
FOR I:=1 TO MDIM DO
  IF(ODD(I)) THEN UV[I]:=MULT(DIVD(PB,MULT(RYND,RYND)),UV[I]);
  ELSE UV[I]:=MULT(DIVD(PB,MULT(RXND,RXND)),UV[I]);
  DATA[1]:=MULT(CBX,DATA[1]); DATA[2]:=MULT(CTX,DATA[2]);
  DATA[3]:=MULT(CBY,DATA[3]); DATA[4]:=MULT(CTY,DATA[4]);
REPEAT (* Until NPROB are completed *)
  IPR:=IPR+1;
  ITER:=ITR[LSELF,IPR];
  IF(NENTRY=0) (* NENTRY=0 for start of GLOBAL PROBLEM *)
    THEN FOR I:=1 TO EDIM DO
      BEGIN DEL[I]:=ZERO; LOAD[I]:=ZERO END
    ELSE FOR I:=1 TO EDIM DO
      BEGIN DEL[I]:=STORED[I+IDX1];LOAD[I]:=STOREL[I+IDX1] END;
ESTIFF(S,EL,DEL,ARND,IXND,IYND,PP,PR,MXP,MYP,MXRE,MYRE);
IDSEC:=NPROB*(LSELF-1)+IPR;
IF IDSEC=1 THEN BEGIN UV[1]:=ZERO; UV[2]:=ZERO; END;
IF IDSEC=NSEC THEN BEGIN UV[MDIM]:=ZERO; UV[MDIM-1]:=ZERO; END;
ZBAR:=DIVD(FLOATI(IDSEC-1),FLOATI(NSEC-1));
UO:=MULT(SINE(MULT(PI,ZBAR)),UINT);
VO:=MULT(SINE(MULT(PI,ZBAR)),VINT);
LOADB[1]:=NEG(PB);
LOADB[2]:=MULT(SUB(ZBAR,ONE),DATA[1]);
LOADB[2]:=SUB(ADD(LOADB[2],UV[IDX2+2]),MULT(ZBAR,DATA[2]));
LOADB[2]:=ADD(MULT(DIVD(PB,MULT(RXND,RXND)),VO),LOADB[2]);
LOADB[3]:=MULT(SUB(ONE,ZBAR),DATA[3]);
LOADB[3]:=ADD(SUB(LOADB[3],UV[IDX2+1]),MULT(ZBAR,DATA[4]));
LOADB[3]:=SUB(LOADB[3],MULT(DIVD(PB,MULT(RYND,RYND)),UO));
TANGT(LOAD,LOADB,DEL,S,CNT2,UO,VO);
TANIDX:=0;
ITR[LSELF,IPR]:=ITR[LSELF,IPR]+1; ITER:=ITR[LSELF,IPR];
IF IO<=3 AND ANY(5) THEN
  BEGIN NXTLN; FOR I:=1 TO EDIM DO
    BEGIN FOR J:=1 TO EDIM DO MSGR(S[I,J]);NXTLN; END; END;
IF IO<=4 AND ANY(5) THEN
  BEGIN MSG('LOAD=');MSGR(PB);MSG(' DET=');MSGR(DET);NXTLN; END;
  IF CMP(DET,TOL2)<=0 OR CNT2=15 THEN TANIDX:=1;
  FOR I:=1 TO 14 DO COEF[I]:=ZERO;
  IF TANIDX<>1 THEN
    BEGIN
      COEF[8]:=SUB(S[2,3],MULT(S[2,1],DIVD(S[1,3],S[1,1])));
      COEF[8]:=MULT(C2,COEF[8]);
      COEF[9]:=MULT(NEG(TWO),COEF[8]);
      COEF[10]:=SUB(MULT(S[2,1],DIVD(S[1,2],S[1,1])),S[2,2]);
    END
  END

```

```

COEF[10]:=MULT(C1,COEF[10]);
COEF[11]:=SUB(MULT(NEG(TWO),COEF[10]),DIVD(PB,MULT(RXND,RXND)));
COEF[12]:=MULT(CBX,SUB(ZBAR,ONE));
COEF[13]:=MULT(CTX,NEG(ZBAR));
COEF[14]:=ADD(PR,ADD(PP,PB));
COEF[14]:=ADD(MULT(PB,DIVD(VO,MULT(RXND,RXND))),
              DIVD(MULT(COEF[14],S[2,1]),S[1,1]));
COEF[14]:=SUB(COEF[14],ADD(MXRE,MXP));
FOR I:=8 TO 14 DO COEF[I]:=NEG(COEF[I]); (* Consistant with tan*)
COEF[1]:=SUB(S[3,3],MULT(S[3,1],DIVD(S[1,3],S[1,1])));
COEF[1]:=MULT(C2,COEF[1]);
COEF[2]:=ADD(MULT(NEG(TWO),COEF[1]),DIVD(PB,MULT(RYND,RYND)));
COEF[3]:=SUB(MULT(S[3,1],DIVD(S[1,2],S[1,1])),S[3,2]);
COEF[3]:=MULT(C1,COEF[3]);
COEF[4]:=MULT(NEG(TWO),COEF[3]);
COEF[5]:=MULT(CBY,SUB(ONE,ZBAR));
COEF[6]:=MULT(CTY,ZBAR);
COEF[7]:=MULT(DIVD(S[3,1],S[1,1]),ADD(ADD(PR,PP),PB));
COEF[7]:=ADD(ADD(COEF[7],MYRE),SUB(MYP,MULT(PB,
              DIVD(UO,MULT(RYND,RYND)))));

FOR I:=1 TO EDIM DO
  BEGIN VCTRD[IDX1+I]:=DEL[I]; VCTRL[IDX1+I]:=LOADB[I]; END;
END;
CLK8:=TREAD;SEND2(NPR+1,IPR,LOCATION(COEF),31);
CLK9:=CLK9+TREAD-CLK8;
IDX1:=IDX1+EDIM;
IDX2:=IDX2+(EDIM-1);
UNTIL (IPR=NPROB) OR TANIDX=1;
CLK8:=TREAD; BAR(7); RECV2(NPR+1,3,LOCATION(NEW),1);
CLK9:=CLK9+TREAD-CLK8;
IF NEW=0 THEN
  FOR I:=1 TO N DO
    BEGIN STORED[I]:=VCTRD[I];STOREL[I]:=VCTRL[I] END;
  UNTIL ANY(6); (* Failure of column *)
  10:NXTLN;
END; (* Routine Ends Here *)
(* ----- MAIN PROGRAM BEGINS ----- *)
BEGIN
  FLGEN(2);FLGEN(5);FLGEN(7);FLGEN(6);
  FLGRES(2);FLGRES(5);FLGRES(7);FLGRES(6);BAR(SYSFLAG);
  ZERO:=CV9512(0.0);
  ONE:=CV9512(1.0);
  TWO:=CV9512(2.0);
  FOUR:=CV9512(4.0);
  TWLVE:=CV9512(12.0);

```

```

SIXTN:=CV9512(16.0);
PI:=CV9512(3.141592654);
(* GET DATA AREAS *)
DALL4::ADDR:=DAPTR(4);
NPROB:=DALL4@[1]; NCNTR:=DALL4@[2];
NB:=DALL4@[3]; ND:=DALL4@[4];
NTB:=DALL4@[5]; NTD:=DALL4@[6];
NSEC:=DALL4@[7]; NPR:=DALL4@[8]; IO:=DALL4@[9];(*NSEC-No.of Sections
NPR -No. of Processors used less one *)
DALL5::ADDR:=DAPTR(5);
B:=CV9512(DALL5@[1]); D:=CV9512(DALL5@[2]); T:=CV9512(DALL5@[3]);
RT:=CV9512(DALL5@[4]); RC:=CV9512(DALL5@[5]);
TOL2:=CV9512(DALL5@[6]);
LENGTH:=CV9512(DALL5@[7]);
KBX:=CV9512(DALL5@[8]); KBY:=CV9512(DALL5@[9]);
KTX:=CV9512(DALL5@[10]); KTY:=CV9512(DALL5@[11]);
UINT:=CV9512(DALL5@[12]); VINT:=CV9512(DALL5@[13]);
E:=CV9512(DALL5@[14]); SIGY:=CV9512(DALL5@[15]);
TOL1:=CV9512(DALL5@[16]);
SIGRC:=NEG(RC);
SIGRT:=RT; NXTLN;
IF CMP(RT,ZERO)=0 THEN MSGLN('MEMBER WITHOUT RESIDUAL STRESSES')
ELSE MSGLN('MEMBER WITH RESIDUAL STRESSES');
(*IF CMP(KBX,ZERO)=0 AND CMP(KTX,ZERO)=0 THEN
    MSGLN('NO PARTIAL RESTRAINT AGAINST X AXIS');
IF CMP(KBY,ZERO)=0 AND CMP(KTY,ZERO)=0 THEN
    MSGLN('NO PARTIAL RESTRAINT AGAINST Y AXIS'); *)
TSTART(1); CLK9:=0;
PROPERTY;
DET:=ONE;
NPROB:=NSEC DIV NPR;
NDIM:=EDIM*NPROB;
MDIM:=(EDIM-1)*NPROB; (* NDIM,MDIM - Variable lengths of STORE,UV *)
ROUTINE(NDIM,MDIM);
TSTOP;
ENDLN(2);MSG('PROCESSING TIME =');MSG(TREAD);NXTLN;
NXTLN;MSG('EXECUTION TIME =');MSG(TREAD-CLK9);NXTLN;
END; (* End of the main program *)

BEGIN (*$NO OBJECT*)
END.

```


APPENDIX D

PROOF-COPY OF THE CONCURRENT PROCESSING PAPER FROM

ENGINEERING WITH COMPUTERS JOURNAL

A proof-copy of the paper mentioned in Section 3.3 is given in this appendix.

PAGES 100 THROUGH 108 INTENTIONALLY OMITTED

APPENDIX E

FRAME SUBSTRUCTURING

E.1 Introduction

The substructuring technique applied for the study presented in Section 4 of this report is given in Reference 9. This appendix presents a summary of the technique.

If a structure such as the one shown in Figure 11 (a) is partitioned into substructures, a restrained structure can be defined by assuming just those joints at the boundaries of the specified substructure to be restrained. Defining the restrained structure in this manner, the various substructures shown in Figure 11 (b) become the basic components to be used in the analysis of the structural system. The advantage of substructuring is that the number of unknown components of joint displacements to be evaluated at any one time in the analysis of the total system is less than the number of unknowns generated by a direct procedure, thus a considerable reduction in CPU time may be realized.

A substructure can be defined as an element or a group of elements that forms a portion of a structural system. The analysis of the substructure with its boundaries fixed can be carried out using the direct procedure as if it were a complete system in itself. From such an analysis, the fixed boundary actions, which are analogous to the fixed end actions for a beam element and the displacements of the unrestrained joints of the substructure stiffness matrix can also be evaluated. Obviously, the joint displacements determined from the analysis will not be true true displacements of the unrestrained joints of the substructure due to the fact that some of the boundaries of the substructure are

PRECEDING PAGE BLANK NOT FILMED

artificially constrained in the basic model, i.e., the restrained structure. Therefore, these displacements must be corrected to account for the response of the substructure to the displacement of its artificially restrained boundaries, which will be determined from the analysis of the total structure. Once the actual displacements of all of the unrestrained joints of the total structure have been evaluated, the final end actions can be determined for each element of the structure.

E.2 Analysis of the Substructure

For the analysis of the k-th substructure, the static joint equilibrium equations can be expressed as follows:

$$[S]\{\Delta\} = \{P\} + \{R\} \quad \text{E.1}$$

The various terms in Equation E.1 are defined as follows:

$[S]$ = stiffness matrix,

$\{\Delta\}$ = displacement vector,

$\{P\}$ = joint load vector, and

$\{R\}$ = boundary reaction vector.

For substructure k, Equation E.1 can be partitioned relative to the unrestrained (u) and restrained (r) boundary displacements of the substructure, as follows:

$$\begin{bmatrix} [S_{uu}] & [S_{ur}] \\ [S_{ru}] & [S_{rr}] \end{bmatrix} \begin{Bmatrix} \{\Delta_u\} \\ \{0\} \end{Bmatrix} = \begin{Bmatrix} \{P_u\} \\ \{P_r\} \end{Bmatrix} + \begin{Bmatrix} \{0\} \\ \{R_r\} \end{Bmatrix} \quad \text{(E.2)}$$

The true response of a substructure, as part of the total system, is the sum of: (i) the response of the substructure, with its boundaries restrained, to the external loads; and (ii) the response of the system to the actual boundary displacements. The equilibrium equation for the response of the substructure due to the displacement of the "restrained"

nodes alone can be written as follows:

$$\begin{bmatrix} [S_{uu}] & [S_{ur}] \\ [S_{ru}] & [S_{rr}] \end{bmatrix} \begin{Bmatrix} \{\Delta_{ud}\} \\ \{\Delta_{rd}\} \end{Bmatrix} = \begin{Bmatrix} \{0\} \\ \{R_{rd}\} \end{Bmatrix} \quad (E.3)$$

in which $\{\Delta_{ud}\}$, and $\{\Delta_{rd}\}$ are, respectively, the displacements of the unrestrained, and restrained nodes of the substructure due to the boundary action vector $\{R_{rd}\}$, required to maintain equilibrium when the boundaries are subjected to the displacement $\{\Delta_{rd}\}$.

Equation E.2 can be expanded as follows:

$$[S_{uu}]\{\Delta_u\} = \{P_u\}, \quad (E.4)$$

$$[S_{ru}]\{\Delta_u\} = \{P_r\} + \{R_r\}. \quad (E.5)$$

Solving Equation E.4 gives:

$$\{\Delta_u\} = [S_{uu}]^{-1}\{P_u\} \quad (E.6)$$

substituting which into Equation E.5 leads to:

$$\{R_r\} = [S_{ru}][S_{uu}]^{-1}\{P_u\} - \{P_r\} \quad (E.7)$$

Similarly, expanding Equation E.3 gives:

$$[S_{uu}]\{\Delta_{ud}\} + [S_{ur}]\{\Delta_{rd}\} = \{0\}, \quad (E.8)$$

$$[S_{ru}]\{\Delta_{ud}\} + [S_{rr}]\{\Delta_{rd}\} = \{R_{rd}\} \quad (E.9)$$

From Equation E.8,

$$\{\Delta_{ud}\} = -[S_{uu}]^{-1}[S_{ur}]\{\Delta_{rd}\} \quad (E.10)$$

substituting which into Equation E.9 leads to:

$$\{R_{rd}\} = [S_s]\{\Delta_{rd}\} \quad (E.11)$$

in which $[S_s]$ is the substructure stiffness matrix given by:

$$[S_s] = [S_{rr}] - [S_{ru}][S_{uu}]^{-1}[S_{ur}] \quad (E.12)$$

E.3 Deflection of Nodes Linking Various Substructures

The static equilibrium of the nodes linking the various substructures $k = 1, 2, 3, \dots, n$, and the real boundaries can be expressed as follows:

$$[S_a]\{\Delta_a\} = \{P_a\} + \{R_a\} \quad (E.13)$$

in which the various elements of $[S_a]$ are found as follows:

$$(S_a)_{ij} = (\sum (S_s)_k)_{ij}, \quad (E.14)$$

where \sum is over k , with $k = 1, 2, 3, \dots, n$. Also, $\{P_a\}$ is the external applied load vector for the nodes linking the various substructures, and $\{\Delta_a\}$ is the true deflection vector for the same nodes. $\{R_a\}$ is the nodal reaction vector.

Equation E.13 can be written in the following form:

$$\begin{bmatrix} [S_{11}] & [S_{12}] \\ [S_{21}] & [S_{22}] \end{bmatrix} \begin{Bmatrix} \{\Delta_{rd}\} \\ \{\Delta_b\} \end{Bmatrix} = \begin{Bmatrix} \{P_{rd}\} \\ \{P_b\} \end{Bmatrix} + \begin{Bmatrix} \{0\} \\ \{R_b\} \end{Bmatrix} \quad (E.15)$$

in which $\{\Delta_{rd}\}$ has been defined in Section E.2. The terms of $\{P_{rd}\}$ are generated using the following equation.

$$(P_{rd})_i = -(\sum (R_{rd})_k)_i + (P)_i, \quad (E.16)$$

where again, \sum is over k , with $k = 1, 2, 3, \dots, n$; and $\{P_b\}$ is the external load vector corresponding to the actual boundaries of the structure. With $\{R_b\} = \{0\}$ Equation E.15 can be expanded to give:

$$\{\Delta_{rd}\} = [S_{11}]^{-1}\{P_{rd}\}, \quad (E.17)$$

$$\{R_b\} = [S_{21}][S_{11}]^{-1}\{P_{rd}\} - \{P_b\}. \quad (E.18)$$

E.4 Deflection of Unrestrained Nodes

The true deflection of the unrestrained nodes is obtained as follows:

$$\{\Delta_{un}\}_k = (\{\Delta_u\}_k + \{\Delta_{ud}\})_k, \quad (E.19)$$

for all k values. Substituting Equation E.10 into Equation E.19 provides:

$$\{\Delta_{un}\}_k = (\{\Delta_u\} - [S_{uu}]^{-1}[S_{ur}]\{\Delta_{rd}\})_k. \quad (E.20)$$

APPENDIX F

FORCE PROGRAM LISTING WITH SAMPLE INPUT/OUTPUT FOR SUBSTRUCTURE ANALYSIS OF 3-STORY FRAME

The information regarding each substructure, the corresponding member properties, and the nodal numbering related to the various members are read from an input file named SSR.IN6. The input variables for this file are SSN, NOM, NOD, NOF, TOR, M, N, MM, NN, L, A, X, where:

SSN = substructure number,

NOM = number of members,

NOD = total number of nodes times the degree of freedom related to each node,

NOF = number of unrestrained nodes times the degree of freedom related to each node,

TOR = type of member (100 for vertical, and 200 for horizontal),

M, N = identification numbers for end 1 of the member,

MM, NN = identification numbers for end 2 of the member,

L = member length,

A = member cross-sectional area,

X = cross-sectional moment of inertia.

The identification numbers for the substructures shown in Figure 11 are as follows. For the upper substructure, the identification numbers M, N, or MM, NN, are:

1, 3 for D,

4, 6 for E,

7, 9 for C, and

10, 12 for F.

The identification numbers for the lower substructure are:

1, 3 for B,
4, 6 for G,
7, 9 for C,
10, 12 for F,
13, 15 for A, and
16, 18 for H.

All members have been assumed to be elastic I-sections, with a Young's modulus of 29,000 ksi, and the following properties:

L = 180.0 in.,
A = 10.3 in.²,
X = 127.4 in.⁴.

The external nodal loads and the connectivity information about the nodes linking the various substructures are read from an input file named SSR.IN7. The input variables for this file are:

LV = the external load vector,
NODD = the number of boundary nodes in the substructure,
NS = node connectivity integer between various substructures (0 if unconnected, and a non-zero integer if connected).

The output of the program is directed to a file named SSR.OPT which stores the degrees of freedom at the global level, and the corresponding deflections.

**ORIGINAL PAGE IS
OF POOR QUALITY**

```

      Force SSR of NP ident VE
* Declaration of arrays and variables
      REAL K11(3,3),K12(3,3),K21(3,3),K22(3,3),_
      Shared DOUBLE PRECISION LV(18),LV1(18),SLRF(18)
      Shared INTEGER TOR, S, TOPS, NOD, NDF, SSN, NOM, F, Z, NS(18), PV(18)
      Shared INTEGER TIME1, TIME2, PVT(2,18)
      Shared DOUBLE PRECISION SM(25,25),DF(40),SUU(18,18),SRR(18,18)
      Shared DOUBLE PRECISION SRU(18,18),DUF(18),RRF(18),SSS(18,18)
      Shared DOUBLE PRECISION RRR(18,18),SS(18,18),DUFF(18)
      Shared DOUBLE PRECISION SS3(21,21),RRF3(21),SUR(18,18)
      Shared DOUBLE PRECISION SURR(18,18),SSSS(2,18,18)
      Shared DOUBLE PRECISION SSUR(2,18,18),SSUU(2,18,18),DDUF(18,2)
      Shared DOUBLE PRECISION SSRR(2,18,18),SSRU(2,18,18)
      CHARACTER*40 FILENM
      End declarations
      Barrier
* ssr.in6-input file for member properties data,ssr.in7-input file for
* external nodal loads,ssr.opt-output file for deflections of the nodes
      open(6,file='/usr/u1/ravi/ssr5.in6',cpu=1)
      open(1,file='/usr/u1/ravi/ssr5.in7',CPU=1)
      open(2,file='/usr/u1/ravi/ssr5.opt',CPU=1)
      CALL OFrtic (TIME1)
      DO 9 I=1,21
      LV1(I)=0.00
      RRF3(I)=0.00
      DO 10 J=1,21
      SS3(I,J)=0.00
10 CONTINUE
9 CONTINUE
      End barrier
      Pcase
      FILENM="/usr/u1/ravi/ssr5.in4"
      CALL FM(1,4,12,6,SSUU,SSUR,SSRU,SSRR,FILENM)
      Usact
      FILENM="/usr/u1/ravi/ssr5.in5"
      CALL FM(2,5,18,6,SSUU,SSUR,SSRU,SSRR,FILENM)
      End pcase
* External nodal loads are read for various substructures by
* different processors
      Barrier
      print*, 'OK'
      NOD=12
      NDF=6
      SSN=1
      F=NOD-NDF
      End barrier
* Equation # 2 is solved using the parallel subroutine 'SOLVE'
124 continue
      Barrier
      READ (1,*) (LV(I),I=1,NDF)
      End barrier

      Presched DO S21 I=1,NDF
      DO S22 C=1,NDF
125 SLL(I,C)=SSLL(SSN,I,C)

```


ORIGINAL PAGE IS
OF POOR QUALITY

```

201 End presched do
Presched do 203 I=1,NOF
DO 204 J=1,F
204 SUR(I,J)=SSUR(SSN,I,J)
203 End presched do
Presched DO 205 I=1,F
DO 206 J=1,NOF
206 SLU(I,J)=SSLU(SSN,I,J)
205 End presched do
Presched do 208 I=1,F
DO 207 J=1,F
SRR(I,J)=SSRR(SSN,I,J)
207 CONTINUE
208 End presched DO

Forcecall SOLVE(SLU,LV,DUF,NOF,PV)
Presched DO 71 I=1,NOF
DDUF(I,SSN)=DUF(I)
PVT(I,SSN)=PV(I)
71 End presched do
* Solving equation number 3 & 4 using parallel subroutine MATMP
Forcecall MATMP(SRU,DUF,RRF,F,NOF,1)
Forcecall SGES2(SUU,SUR,SSS,NOF,F,PV)
Forcecall MATMP(SRU,SSS,RRR,F,NOF,F)

Presched do 117 I=1,NOF
DO 118 J=1,F
SSSS(SSN,I,J)=SSS(I,J)
118 CONTINUE
117 End presched do
Presched DO 81 I=1,F
DO 82 J=1,F
SS(I,J)=SRR(I,J)-RRR(I,J)
82 CONTINUE
81 End presched do
* Forming SRR effective for every substructure and placing the elements
* of substructure
Barrier
READ(6,*) NODD
PRINT*,NODD,'NODD OK'
READ (6,*) (NS(I),I=1,NODD)
PRINT*,NS(NODD),'NS OK'
NC=1
DO 120 I=1,NODD
IF(NS(I).EQ.0) GOTO 120
RRF3(NC)=RRF3(NC)+RRF(NC)
NC=1
DO 121 J=1,NODD
IF(NS(J).EQ.0) GOTO 121
SS3(I,J)=SS3(I,J)+SS(NC,NC)
NC=NC+1
121 CONTINUE
NC=NC+1
120 CONTINUE
SSN=2

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

NOD=18
NOF=6
End barrier
IF(SSN.EQ.1) GOTO 104
Barrier
SSN=3
NOD=12
NOF=6
F=NOD-NOF
DO 49 I=1,NOF
LV1(I)=(-1.0)*RRF3(I)
DO 50 J=1,NOF
SUU(I,J)=SS3(I,J)
50 CONTINUE
49 CONTINUE
READ (1,*) (LV(I),I=1,NOF)
DO 139 I=1,NOF
LV(I)=LV(I)+LV1(I)
139 CONTINUE
print*, 'OK'
End barrier
Forcecall SOLVE(SUU, LV, DUF, NOF, PV)
Forcecall MATM2(SSSS, DUF, DUFF, NOF, F, 1)
Barrier
print*, 'OK TILL HERE'
* Solving equation 6 for substructure number 1
DO 69 I=7,12
DUF(I)=DDUF(I-6,1)-DUFF(I-6)
69 CONTINUE
End barrier
Forcecall MATM2(SSSS, DUF, DUFF, NOF, F, 2)
Barrier
print*, 'OK'
* Solving equation 6 for substructure number 2
DO 79 I=13,18
DUF(I)=DDUF(I-12,2)-DUFF(I-12)
79 CONTINUE
CALL CFrtic (TIME2)
print*, 'OK'
print*, TIME2, TIME1
* Writing deflections for common nodes using single processor
WRITE (2,78)
WRITE (2,75)
78 FORMAT (/,5X,'DEFLECTIONS OF NODES COMMON TO SUBSTRUCTURES 1 & 2',
/,/)
75 FORMAT (5X,'DEG. OF FREEDOM',16X,'DEFLECTION',/)
77 FORMAT (10X,12,15X,1P12.5)
DO 76 I=1,6
WRITE (2,77) 1,DUF(I)
WRITE (2,*)
76 CONTINUE
WRITE (2,69)
WRITE (2,75)
* Writing deflections of nodes of substructure 1 using one processor
85 FORMAT (/,5X,'DEFLECTIONS OF NODES OF SUBSTRUCTURE 1',/)

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

DO 100 I=7,12
WRITE (2,77) I,DUF(I)
WRITE (2,*)
100 CONTINUE
* Writing deflections of nodes of substructure II using single processor
111 FORMAT (/,5X,'DEFLECTIONS OF NODES OF SUBSTRUCTURE II'/)
WRITE (2,111)
WRITE (2,75)
DO 109 I=13,18
WRITE (2,77) I,DUF(I)
WRITE (2,*)
109 CONTINUE
End Barrier
Join
END
SUBROUTINE FM(SSN,NOM,NOD,NDF,SSUU,SSUR,SSRU,SSRR,FILENM)
REAL L,K11(3,3),K12(3,3),K21(3,3),K22(3,3)
INTEGER TOR,SSN,F,Z
DOUBLE PRECISION LV(18),LV1(18)
DOUBLE PRECISION SSUU(2,18,18), SSUR(2,18,18),SM(18,18)
DOUBLE PRECISION SSRU(2,18,18),SSRR(2,18,18)
CHARACTER*40 FILENM
OPEN (4,FILE=FILENM,CPU=1)
E=29000.0
* Sub-structure stiffness matrix formation starts.
* NOD-Total degrees of freedom in the substructure,NDF-total number of interior
* nodes,NOM=number of members,SSN=number of the substructure
DO 11 I=1,NOD
DO 12 J=1,NOD
SM(I,J)=0.0
12 CONTINUE
11 CONTINUE
* Reads individual member properties,L=length,A=area of cross-section
* X-inertia of the section
* Reads M,N-leftside DOF numbering,MM,NN-right side DOF numbering,
* TOR is 100 for vertical members,200 for horiz. members
DO 108 Z=1,NOM
READ(4,*) L,A,X
READ(4,*) M,N,MM,NN,TOR
print*, 'M,N,OK'
DO 17 I=1,3
DO 16 J=1,3
K11(I,J)=0
K12(I,J)=0
K21(I,J)=0
K22(I,J)=0
16 CONTINUE
17 CONTINUE
IF (TOR.EQ.100) GO TO 110
K11(1,1)=12*E*X/L**3
K11(1,3)=(6*E*X/L**3)*(-1)
K11(3,1)=K11(1,3)
K11(3,3)=E*A/L
K11(3,1)=K11(1,3)

```

ORIGINAL PAGE IS
OF POOR QUALITY

```
K11(3,2)=K11(2,3)
K11(3,3)=4*E*X/L
K12(1,1)=K11(1,1)*(-1)
K12(1,2)=K11(1,2)
K12(1,3)=K11(1,3)
K12(2,1)=K11(1,2)
K12(2,2)=K11(2,2)*(-1)
K12(2,3)=K11(2,3)
K12(3,1)=K12(1,3)*(-1)
K12(3,3)=2*E*X/L
DO 19 I=1,3
DO 20 J=1,3
K21(I,J)=K12(J,I)
20 CONTINUE
19 CONTINUE
K22(1,1)=12*E*X/L**3
K22(1,3)=6*E*X/L**2
K22(2,2)=E*A/L
K22(3,1)=6*E*X/L**2
K22(3,3)=4*E*X/L
245 KK=0
LL=0
DO 21 I=MM,NN
KK=KK+1
DO 22 J=M,N
LL=LL+1
SM(I,J)=SM(I,J)+K21(KK,LL)
22 CONTINUE
LL=0
21 CONTINUE
CONTINUE
KK=0
LL=0
DO 23 I=MM,NN
KK=KK+1
DO 24 J=MM,NN
LL=LL+1
SM(I,J)=SM(I,J)+K22(KK,LL)
24 CONTINUE
LL=0
23 CONTINUE
KK=0
LL=0
DO 25 I=M,N
KK=KK+1
DO 26 J=MM,NN
LL=LL+1
SM(I,J)=SM(I,J)+K12(KK,LL)
26 CONTINUE
LL=0
25 CONTINUE
KK=0
LL=0
DO 27 I=M,N
KK=KK+1
```

ORIGINAL PAGE IS
OF POOR QUALITY

```

DO 26 J=M, N
LL=LL+1
SM(I, J)=SM(I, J)+K11(KK, LL)
26 CONTINUE
LL=0
27 CONTINUE
GOTO 108
110 K11(1, 1)=E*A/L
K11(2, 2)=12*E*X/L**3
K11(2, 3)=6*E*X/L**2
K11(3, 2)=K11(2, 3)
K11(3, 3)=4*E*X/L
K12(1, 1)=K11(1, 1)*(-1)
K12(2, 2)=K11(2, 2)*(-1)
K12(2, 3)=K11(2, 3)
K12(3, 2)=K11(2, 3)*(-1)
K12(3, 3)=2*E*X/L
DO 29 I=1, 3
DO 30 J=1, 3
K21(I, J)=K12(J, I)
30 CONTINUE
29 CONTINUE
K22(1, 1)=K11(1, 1)
K22(2, 2)=K11(2, 2)
K22(2, 3)=K12(3, 2)
K22(3, 2)=K12(3, 2)
K22(3, 3)=K11(3, 3)
240 KK=0
LL=0
DO 31 I=M, N
KK=KK+1
DO 32 J=MM, NN
LL=LL+1
SM(I, J)=SM(I, J)+K12(KK, LL)
32 CONTINUE
LL=0
31 CONTINUE
KK=0
LL=0
DO 33 I=M, N
KK=KK+1
DO 34 J=M, N
LL=LL+1
SY(I, J)=SY(I, J)+K11(KK, LL)
34 CONTINUE
LL=0
33 CONTINUE
KK=0
LL=0
DO 35 I=MM, NN
KK=KK+1
DO 36 J=MM, NN
LL=LL+1
SY(I, J)=SY(I, J)+K22(KK, LL)
36 CONTINUE

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

35      LL=0
      CONTINUE
      KK=0
      LL=0
      DO 37 I=MM, NN
      KK=KK+1
      DO 38 J=M, N
      LL=LL+1
      SM(I, J)=SM(I, J)+K21(KK, LL)
38      CONTINUE
      LL=0
37      CONTINUE
108     CONTINUE
* Sub structure stiffness matrix formation ends for various
* sub structures.
* Matrix for various sub structures is divided according
* to number of interior and exterior nodes
      F=NOD-NOF
      DO 1 I=1, NOF
      DO 2 J=1, NOF
      SSUU(SSN, I, J)=SM(I, J)
2      CONTINUE
1      CONTINUE
      DO 3 I=1, F
      DO 4 J=1, F
      SSRR(SSN, I, J)=SM(I+NOF, J+NOF)
4      CONTINUE
3      CONTINUE
      DO 5 I=1, NOF
      DO 6 J=1, F
      SSUR(SSN, I, J)=SM(I, J+NOF)
6      CONTINUE
5      CONTINUE
      DO 7 I=1, F
      DO 8 J=1, NOF
      SSRU(SSN, I, J)=SM(I+NOF, J)
8      CONTINUE
7      CONTINUE
      RETURN
      END
* Subroutine for inversion of matrix (sequential)
      SUBROUTINE INV(A, N)
      DOUBLE PRECISION A
      DIMENSION A(18, 18)
      NP=N+1
      A(1, NP)=1
      DO 10 K=2, N
10      A(K, NP)=0
      DO 40 J=1, N
      DO 20 L=1, N
20      A(NP, L)=A(1, L+1)/A(1, 1)
      DO 30 K=2, N
      DO 30 L=1, N
10      A(K+1, L)=A(K, L+1)-A(K, 1)*A(NP, L)
      DO 40 L=1, N

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

42  A(N,L)=A(NR,L)
    RETURN
    END
* Subroutine for matrix multiplication
  SUBROUTINE MULT(A,B,C,M,N)
    DOUBLE PRECISION A(18,18),B(18),C(18)
    DO 30 I=1,M
      C(I)=0.0
      DO 30 L=1,N
30    C(I)=C(I)+A(I,L)*B(L)
      RETURN
    END
* Subroutine for matrix multiplication (sequential)
  SUBROUTINE MULT2(A,B,C,M,K,N)
    DOUBLE PRECISION A(18,18),B(18,18),C(18,18)
    DO 20 J=1,N
      DO 30 I=1,M
        SUM=0.0
        DO 40 L=1,K
          SUM=SUM+A(I,L)*B(L,J)
40    CONTINUE
        C(I,J)=SUM
30    CONTINUE
20    CONTINUE
      RETURN
    END
  SUBROUTINE MULT3(A,B,C,M,K,N,II)
    DOUBLE PRECISION A(2,18,18),B(2,18,18),C(18,18)
    DO 20 J=1,N
      DO 30 I=1,M
        SUM=0.0
        DO 40 L=1,K
          SUM=SUM+A(II,I,L)*B(II,L,J)
40    CONTINUE
        C(I,J)=SUM
30    CONTINUE
20    CONTINUE
      RETURN
    END
* Force subroutine for matrix multiplication in parallel
  Forcesub MATXP(A,B,C,M,N,D) of NO ident VE
  DOUBLE PRECISION A(18,18),B(18,18),C(18,18)
  INTEGER M,N,D
  Shared INTEGER LDIM
  Private INTEGER I,J,K
  Private DOUBLE PRECISION SUM
  Enc declarations
  Barrier
  Enc Barrier
  Presched DO 300 I=1,M
    DO 200 J=1,N
      SUM=0.0
      DO 100 K=1,N

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

        SUM=SUM+A(I,K)*B(K,J)
100  CONTINUE
    C(I,J)=SUM
200  CONTINUE
300  End presched do
    Barrier
    End barrier
    RETURN
    END
* Force subroutine for solving equation # 7
    Forcesub MATM2(A,B,C,M,N,II) of NP ident MP
    DOUBLE PRECISION A(2,18,18),B(18),C(18)
    INTEGER M,N,II
    Private INTEGER I,K
    Private DOUBLE PRECISION SUM
    End Declarations
    Barrier
    End Barrier
    Presched DO 300 I=1,M
        SUM=0.0
        DO 100 K=1,N
            SUM=SUM+A(II,I,K)*B(K)
100  CONTINUE
        C(I)=SUM
300  End presched do
    Barrier
    End Barrier
    RETURN
    END
* Force subroutine for solving [A][x]=[b] in parallel
    Forcesub SOLVE(A,B,X,N,IPVT) of NP ident ME
C.....Test program for parallel PLU decomposition using Force versions
C.....of "Linpack" routines.
C.....This program calls a (single stream) routine to read in a matrix
C.....and vector, then SGEFA is called to factor the matrix into a PLU form,
C.....then SGESL is called to solve the system. The computation is timed.
C.....and then a (single stream) routine is called to output the results.

    INTEGER N,IPVT(18)
    Shared INTEGER INFO
    DOUBLE PRECISION A(18,18), B(18),X(18)
    Shared INTEGER ITM1, ITM2, ITM3
    REAL DELTA1, DELTA2, RATE1, RATE2
    End declarations

    Barrier
    CALL CFrtic(ITM1)
    End barrier

    Forcesubcall SGEFA(A, 18, N, IPVT, INFO)
    IF (INFO.NE. 0) THEN
1      matrix is singular...
        IF (me .eq. 1) print *, 'matrix is singular'
        Goto

```


ORIGINAL PAGE IS
OF POOR QUALITY

```

END IF

IF (ME.EQ. 1) CALL CFrtic(ITM2)

Forcescall SGESL(A, 18, N, IPV, B, 0)
Barrier
DO 44 I=1,N
44 X(I)=B(I)
CALL CFrtic(ITM3)
CALL PRINTB(A, B, N)
do 25 i=1,n
25 print*, (a(i,j),j=1,n),o(i),x(i)

DELTA1 = (ITM2 - ITM1)/50.0
DELTA2 = (ITM3 - ITM2)/50.0
IF (DELTA1.NE. 0) THEN
    RATE1 = (2.0*N*N*N/3.0)*1.0E-6/DELTA1
    WRITE (*, 400) NP, N, DELTA1, RATE1
ELSE
    WRITE (*, 400) NP, N, DELTA1, 0.0
    WRITE (*, 402)
ENDIF
IF (DELTA2.NE. 0) THEN
    RATE2 = (N*N)*1.0E-6/DELTA2
    WRITE (*, 401) DELTA2, RATE2
ELSE
    WRITE (*, 401) DELTA2, 0.0
    WRITE (*, 402)
ENDIF
400 FORMAT(' Number of processes =',I3,' Matrix order =',I4, /
+ ' matrix decomposition: ',F12.4,' seconds (' ,F5.3,' MFLOPS)')
401 FORMAT(
+ ' Fwd/back subst: ',F12.4,' seconds (' ,F5.3,' MFLOPS)')
402 FORMAT(' No elapsed time measured.')
End barrier
RETURN
END

*****
Forcesub SGESL(A, LDA, N, IPV, B, JOB) of NP ident me
SGESL: Force parallel version of LINPACK's SGESL which solves
Ax=b, using (column orientated) forward/backward subst.
A call to SGEFA must precede the call to SGESL.
SGEFA provides the LU factorization of A.
A: the working matrix
LDA: the declared dimension of A
N: the working dimension of A
IPVT: the pivot array
B: the input/output vector (solved in place)
B is the only argument altered by SGESL.
JOB: job selector: JOB=0: solves Ax=b
JOB=1: solves Transpose(A)x=b, not implemented
All argument MUST be declared "Shared" in the calling module.
integer LDA, N, IPV(18), JOB
DOUBLE PRECISION A(LDA,1), B(1)
Private integer I, K,

```

```

Shared DOUBLE PRECISION TEM
End declarations

C      first solve:  L*V = B
do 20 K = 1, N-1
    Barrier
    L      = IPVT(K)
    TEM    = B(L)
    B(L)   = B(K)
    B(K)   = TEM
    End barrier
    Presched do 22  I = K+1, N
        B(I) = B(I) + TEM * A(I,K)
22      End presched do
20      continue

C      then solve:  U*X = Y
do 40 K = N, 1, -1
    Barrier
    B(K)   = B(K)/A(K,K)
    TEM    = -B(K)
    End barrier
    Presched do 42  I = 1, K-1
        B(I) = B(I) + TEM * A(I,K)
42      End presched do
40      continue
    Barrier
    End barrier
    RETURN
END

```

```

Forcesub SGEFA(A, LDA, N, IPVT, INFO) of NP ident ME
INTEGER LDA, N, IPVT(LDA), INFO
DOUBLE PRECISION A(LDA, LDA)

```

```

C
Private DOUBLE PRECISION ABSMAX
Private DOUBLE PRECISION T
Private INTEGER KK, I, MAXI, J, KP1
Shared INTEGER IALL
Shared DOUBLE PRECISION PIV, ALLMAX
Shared logical ILOCK
End declarations

```

```

Barrier
INFO = 0
ALLMAX = 0.0
End barrier

```

C.....For each row of the matrix

```

DO 1000 KK = 1, N-1

IF(INFO.NE.0) GOTO 2000

```

C.....Reset local maxima

ORIGINAL PAGE IS
OF POOR QUALITY

ABSMAX = 0.0

B.....Find the pivot row

```
Presched DO 100 I = KK, N
  T = ABS( A(I, KK) )
  IF (ABSMAX .LT. T) THEN
    MAXI = I
    ABSMAX = T
  ENDIF
100 End Presched DO
```

C.....Update the Shared Information if necessary

```
Critical ILOCK
IF (ALLMAX .LT. ABSMAX) THEN
  IALL = MAXI
  ALLMAX = ABSMAX
ENDIF
End critical
```

```
Barrier
IF (ALLMAX .EQ. 0.0) THEN
  INFO = KK
  IPVT(KK)=KK
ELSE
  IPVT(KK)=IALL
ENDIF
End Barrier
```

C.....If the matrix is singular then pass the information

```
IF (INFO .EQ. KK) GOTO 1000
MAXI = IALL
IF (MAXI .NE. KK) THEN
```

C.....Swap rows if necessary

```
Presched DO 110 J = KK, N
  TEMP = A(MAXI, J)
  A(MAXI, J) = A(KK, J)
  A(KK, J) = TEMP
110 End Presched DO
ENDIF
```

D.....Self schedule row reductions

```
KP1 = KK - 1
Barrier
PIV = -1.0/A(KK, KK)
End Barrier
```

```
Selfsched DO 130 I = KP1, N
  TEMP = PIV*A(I, KK)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```

      A(I, KK) = TEMP
      DO 120 J = KP1, N
        A(I, J) = A(I, J) + TEMP*A(KK, J)
120    CONTINUE
130    End Selfsched DO

      Barrier
      ALLMAX=0
      End barrier

1000  CONTINUE

2000  RETURN
      END

      Forcesub SBES2(A, B, X, N, M, IPVT) of NP ident ME
      INTEGER N, IPVT(18), JOB, M
      DOUBLE PRECISION A(18,18), B(18,18), X(18,18)
      Private integer I, K, L, J
      Shared DOUBLE PRECISION TM
      End declarations
      DO 30 J=1, M
*first solve L*Y=B
        DO 20 K=1, N-1
          Barrier
          L=IPVT(K)
          TM=B(L, J)
          B(L, J)=B(K, J)
          B(K, J)=TM
          End barrier
          Presched do 22 I=K+1, N
            B(I, J)=B(I, J)+TM*A(I, K)
22        End presched do
20      continue
* then solve U*X=y
        DO 40 K=N, 1, -1
          Barrier
          B(K, J)=B(K, J)/A(K, K)
          TM=-B(K, J)
          End barrier
          Presched do 42 I=1, K-1
            B(I, J)=B(I, J)+TM*A(I, K)
42        End presched do
40      continue
        do 25 I=1, M
25      X(I, J)=B(I, J)
30    continue
      Barrier
      End barrier
      RETURN
      END

      Forcesub SBES3(A, B, C, N, M, IPVT(18)) of NP ident ME
      INTEGER N, M, IPVT(18,2), JOB
      Private INTEGER I, J, K, L
      DOUBLE PRECISION A(2,18,18), B(2,18,18), C(18,18)

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

Share@9008LE PRECISION TN
End calculations
DO 30 J=1,NJ=
* FIRST SOLVE BI*y=B
DO 20 K=1,N-1
Barrier
L=IPVT(K,II)
TN=B(II,L,J)
B(II,L,J)=B(II,K,J)
B(II,K,J)=TN
End barrier
Presched DO 22 I=K+1,N
B(II,I,J)=B(II,I,J)+TN*A(II,I,K)
22 End Presched do
20 CONTINUE
*NOW SOLVE U*X=Y
DO 40 K=N,1,-1
Barrier
B(II,K,J)=B(II,K,J)/A(II,K,K)
TN=-B(II,K,J)
End barrier
Presched do 42 I=1,K-1
B(II,I,J)=B(II,I,J)+TN*A(II,I,K)
42 End Presched do
40 CONTINUE
DO 50 I=1,N
C(I,J)=B(II,I,J)
50 CONTINUE
30 CONTINUE
RETURN
END

```

JOB NAME = ssr.in6

ORIGINAL PAGE IS
OF POOR QUALITY

ssr.in6

12, 6, 4, 1
180.0, 10.3, 127.0
1, 3, 4, 6, 200
180.0, 10.3, 127.0
7, 9, 10, 12, 200
180.0, 10.3, 127.0
7, 9, 1, 3, 100
180.0, 10.3, 127.0
10, 12, 4, 6, 100
12
1, 2, 3, 4, 5, 6, 0, 0, 0, 0, 0, 0
18, 6, 5, 2
180.0, 10.3, 127.0
1, 3, 4, 6, 200
180.0, 10.3, 127.0
1, 3, 7, 9, 100
180.0, 10.3, 127.0
4, 6, 10, 12, 100
180.0, 10.3, 127.0
13, 15, 1, 3, 100
180.0, 10.3, 127.0
16, 18, 4, 6, 100
12
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
7 [3]

JOB NAME = ssr.in7

ssr.in7

00.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
00.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
0 [\$]

ORIGINAL PAGE IS
OF POOR QUALITY

JOB NAME = ssr.oop

ORIGINAL PAGE IS
OF POOR QUALITY

ssr3.oop

DEFLECTIONS OF NODES COMMON TO SUBSTRUCTURES I & II

DEG. OF FREEDOM	DEFLECTION
1	11.72159
2	0.06937
3	-0.02147
4	11.71558
5	-0.06937
6	-0.02145

DEFLECTIONS OF NODES OF SUBSTRUCTURE I

DEG. OF FREEDOM	DEFLECTION
7	15.88023
8	0.07696
9	-0.01008
10	15.87420
11	-0.07696
12	-0.01009

DEFLECTIONS OF NODES OF SUBSTRUCTURE II

DEG. OF FREEDOM	DEFLECTION
13	4.89344
14	0.04478
15	-0.02506
16	4.89343
17	-0.04478
18	-0.02504

APPENDIX G

TITLES OF ABSTRACTS FOR 29TH AIAA/SDM CONFERENCE

The following abstracts are submitted for the 29th AIAA/SDM Conference to be held in Williamsburg, Virginia, April 18-20, 1988:

1. Prasad, V., Razzaq, Z., and Storaasli, O.O., "Two-Dimensional Concurrent Finite Element Analysis of Rectangular Panel with Hole."
2. Bhati, R., Razzaq, Z., and Storaasli, O.O., "Concurrent Substructure Analysis of Plane Frames."